

Risk Analysis and Measurement with CWRAF

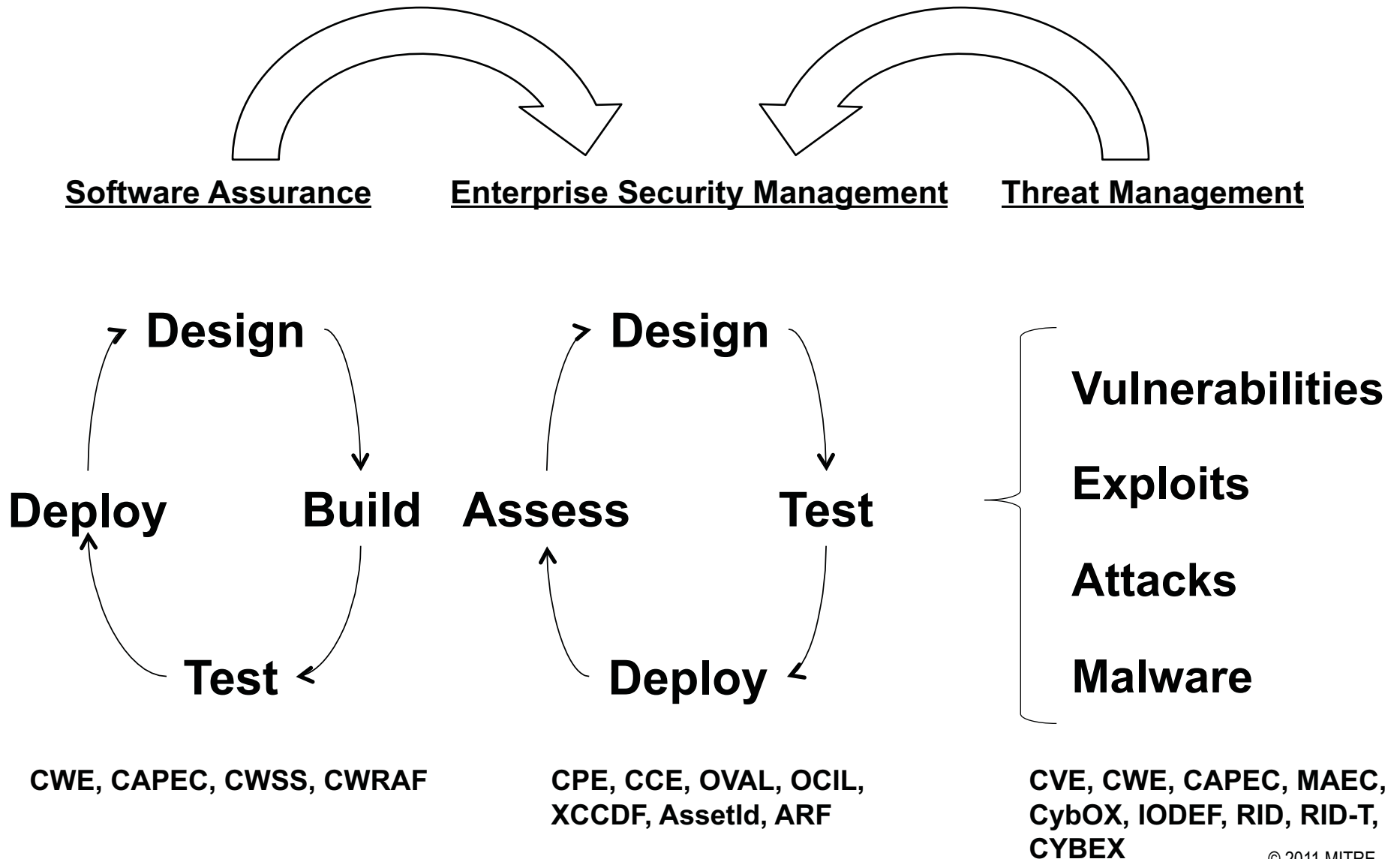
October 25, 2011

Robert A. Martin



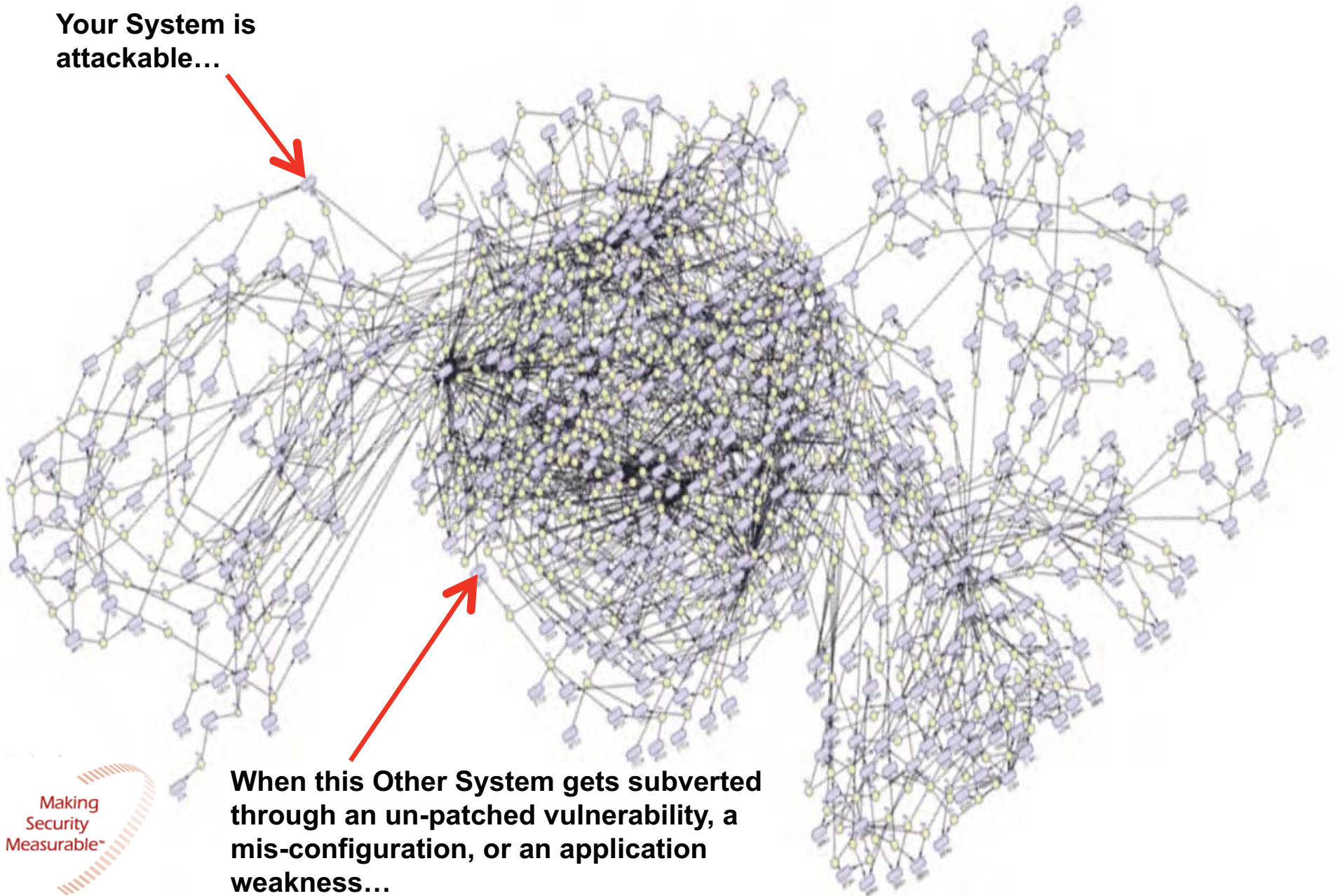
Making Security Measurable (MSM)

“You Are Here”



Today Everything's Connected – Like an Ecosystem

Your System is
attackable...



Making
Security
Measurable™

ECOSYSTEM

SOLAR RADIATION

MOISTURE

DISTURBANCES

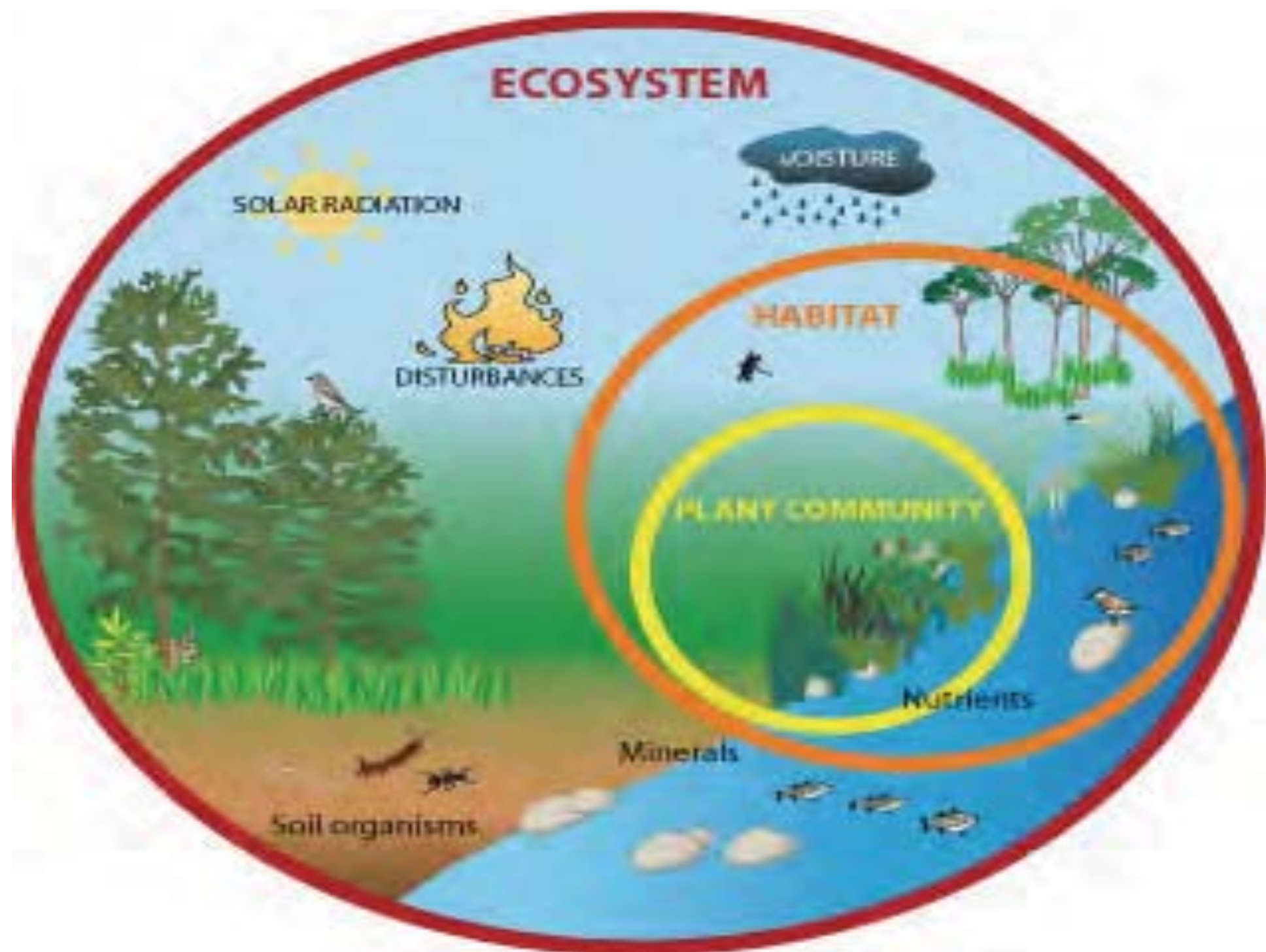
HABITAT

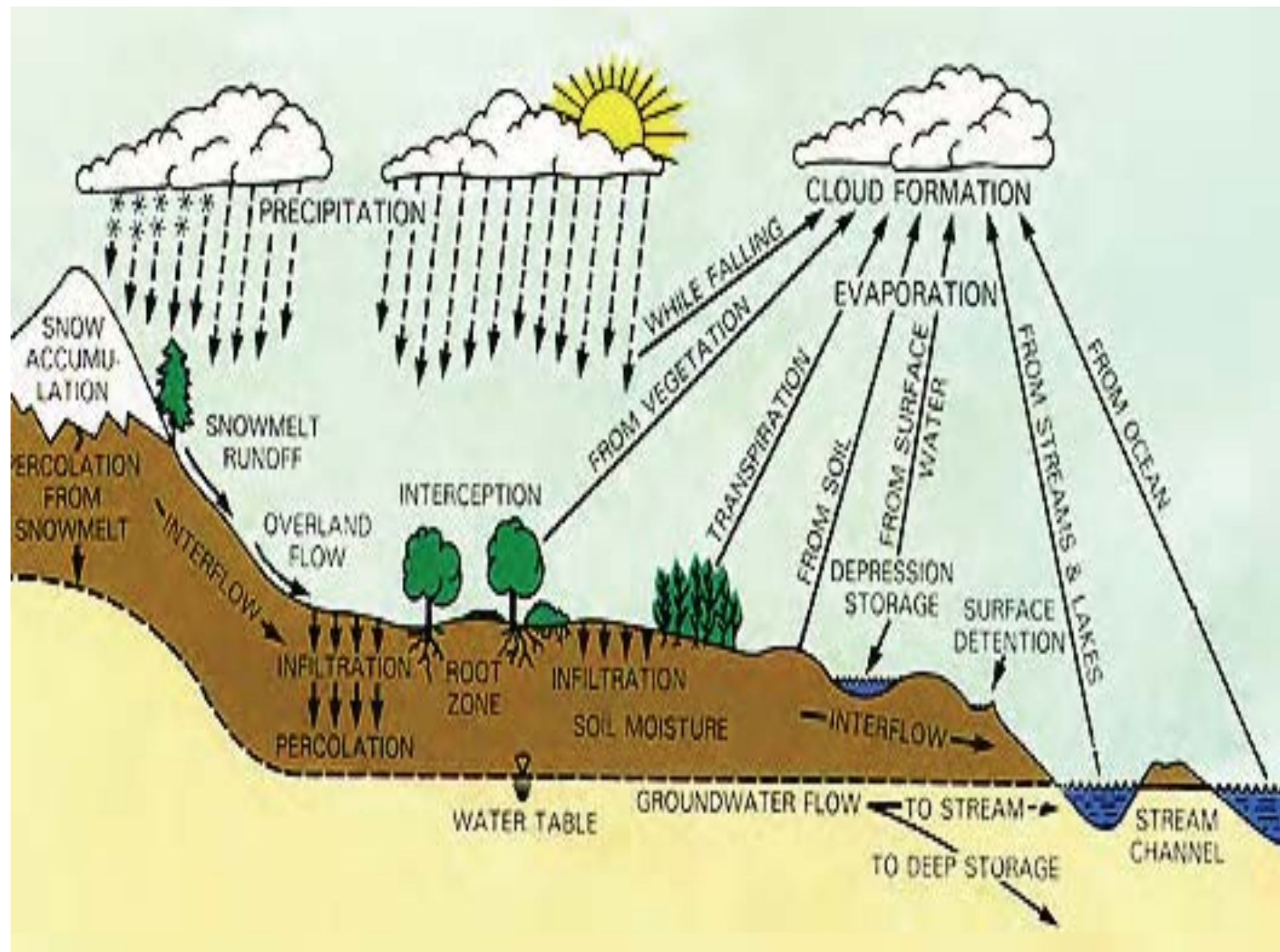
PLANT COMMUNITY

Nutrients

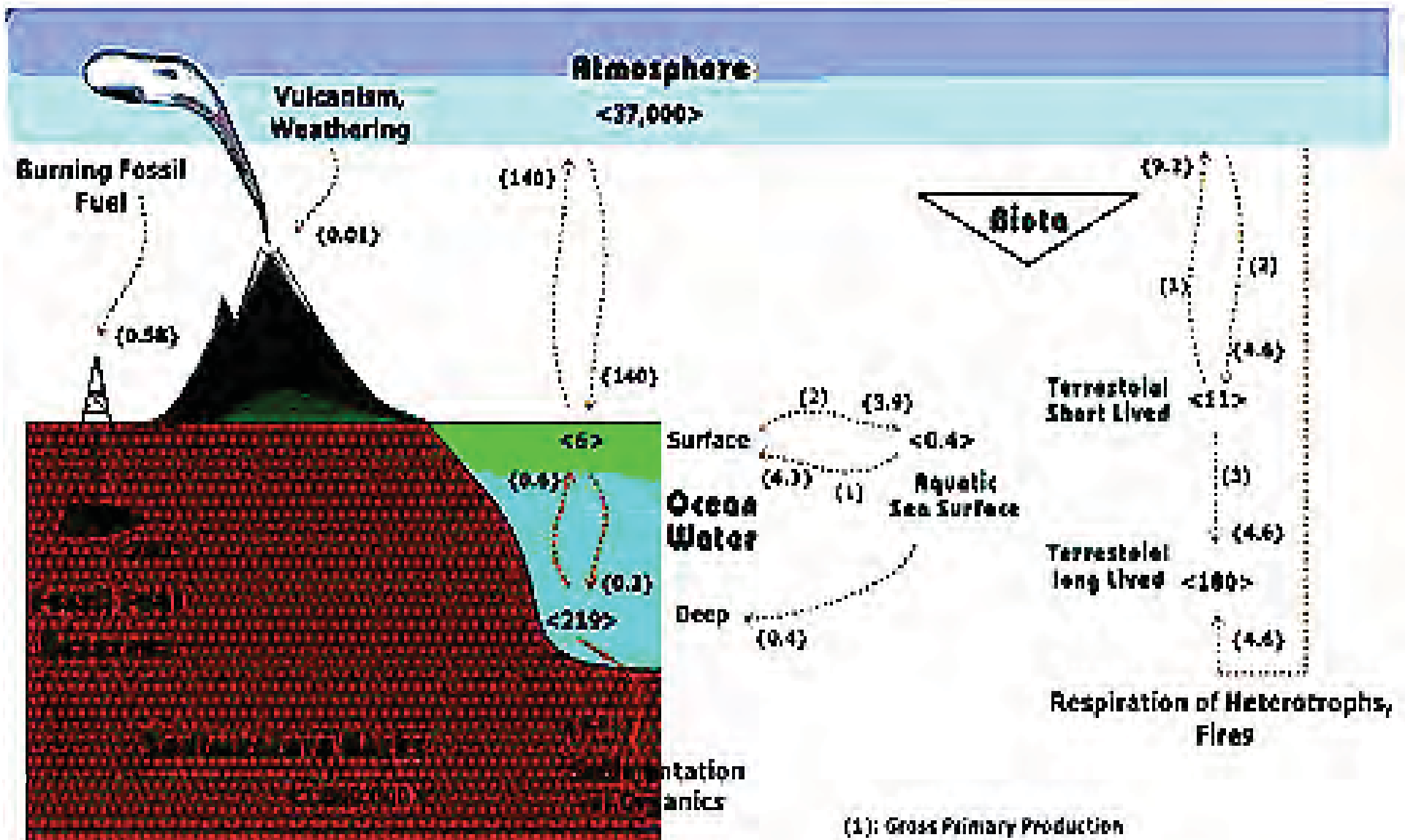
Minerals

Soil organisms





Global Oxygen Cycling on Earth



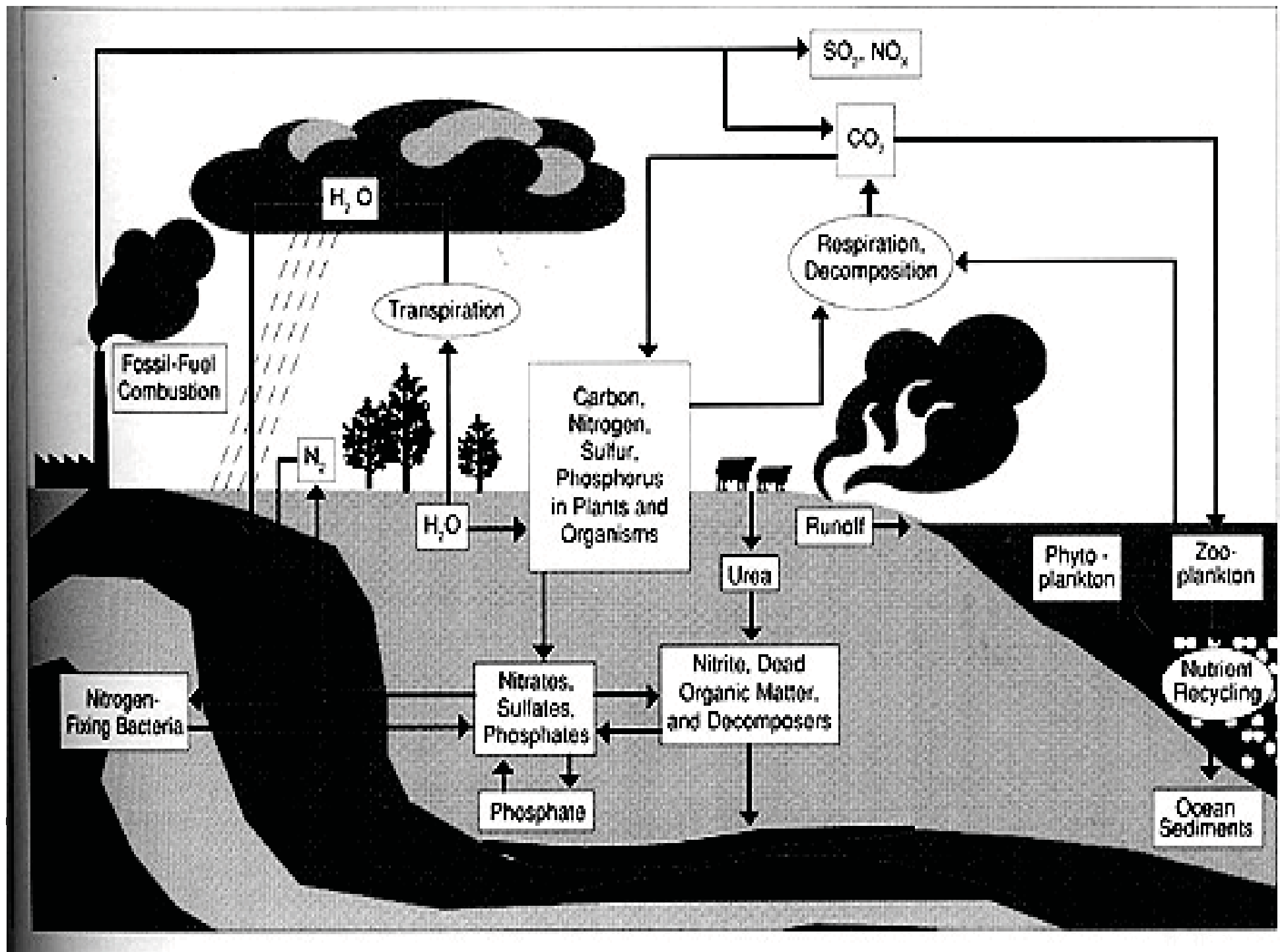
Oxygen (pool sizes) 10^{15} moles

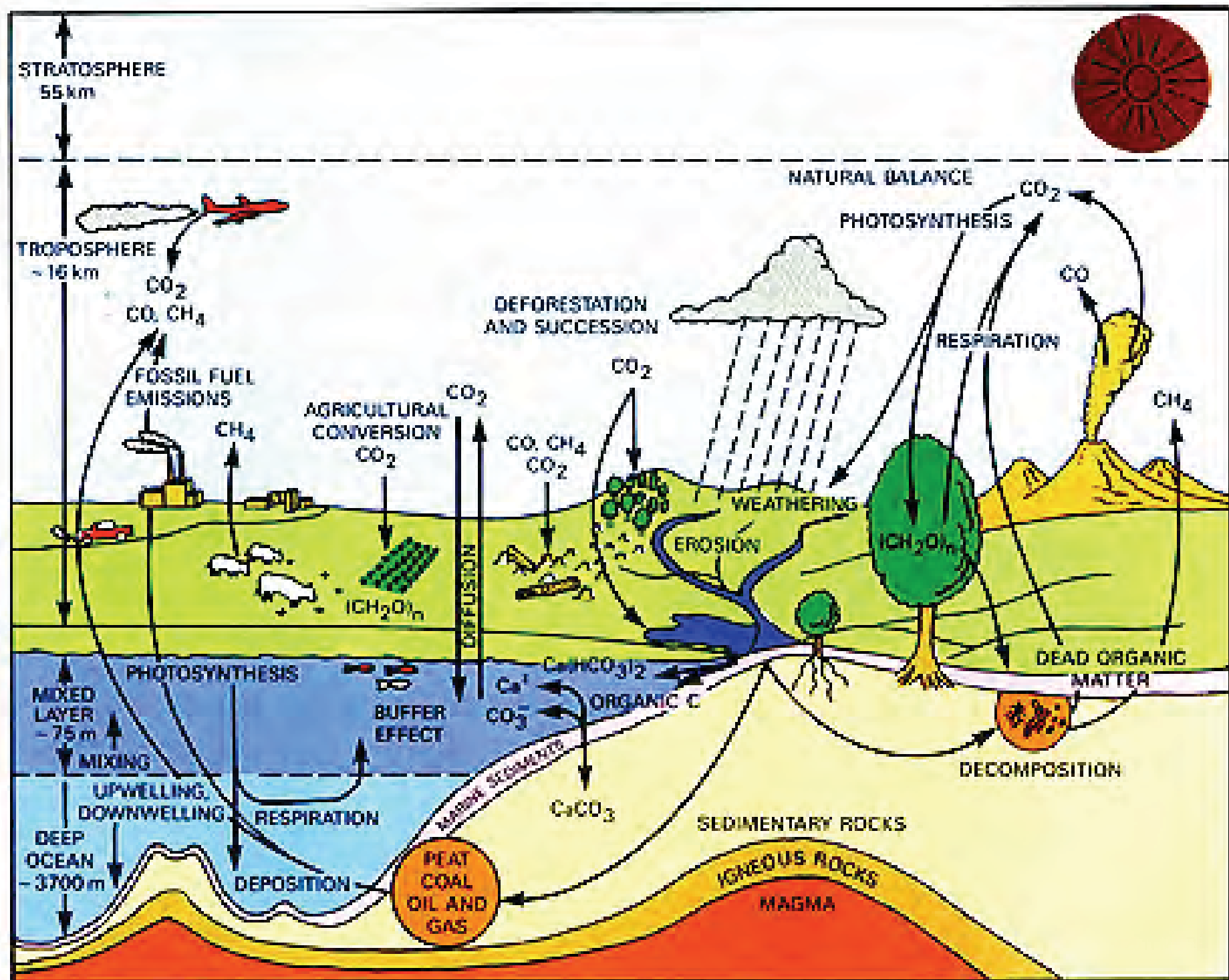
Oxygen (pool fluxes) 10^{14} moles / year

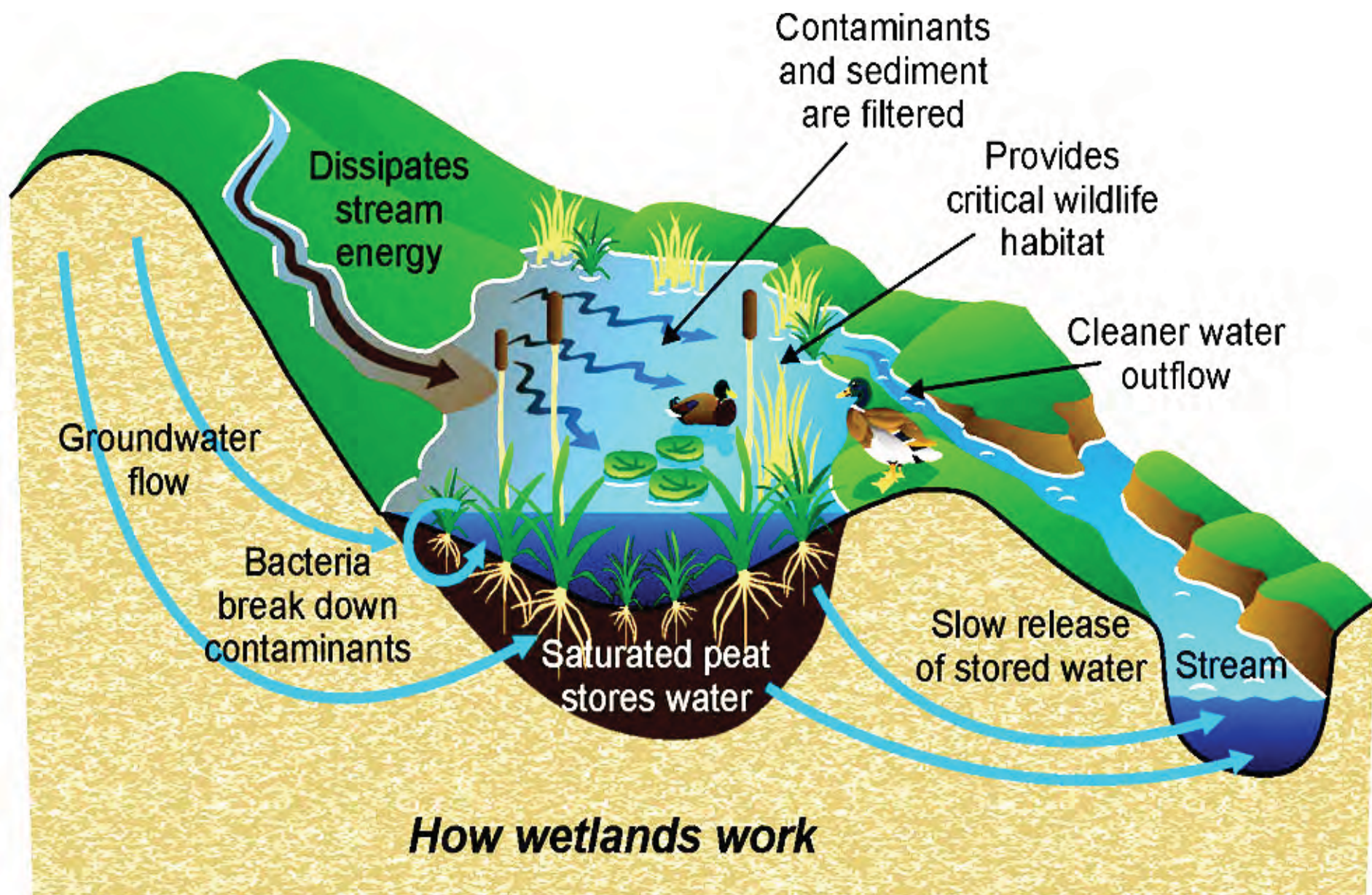
(1): Gross Primary Production

(2): Net Respiration

(3): Net Primary Production (= 1-2)







Linkage with Fundamental Changes in Enterprise Security Initiatives

Enabling Distributed Security in Cyberspace

Building a Healthy and Resilient Cyber Ecosystem with Automated Collective Action

March 23, 2011

- **Technical Interoperability.** The ability for different technologies to communicate and exchange data based upon well defined and widely adopted interface standards.
- **Policy Interoperability.** Common business processes related to the transmission, receipt, and acceptance of data among participants.

Within cybersecurity, all three types of interoperability are being enabled through an approach that has been refined over the past decade by many in industry, academia, and government. It is an information-oriented approach, generally referred to as [cyber] *security content automation* and comprises the following elements.¹³

- **Enumerations.** These are lists or catalogs of the fundamental entities of cybersecurity, for example, cyber devices and software items (CPE); device and software configurations (CCE); publicly known weaknesses in architecture, design, or code (CWE); publicly known flaws or vulnerabilities (CVE); or publicly known attack patterns (CAPEC). Enumerations enable semantic interoperability.
- **Languages and Formats.** These incorporate enumerations and support the creation of machine-readable security state assertions, assessment results, audit logs, messages, and reports. Examples include patterns associated with assets, configurations, vulnerabilities, and software patches (XCCDF & OVAL); security announcements (CAIF), events (CEE), malware (MAEC); risk associated with vulnerability (CVSS), sensor collection and correlation (ARF), and US-CERT security bulletins and incident reports (NIEM). Languages and formats enable technical interoperability.
- **Knowledge Repositories.** These contain a broad collection of best practices, benchmarks, profiles, standards, templates, checklists, tools, guidelines, rules, and principles, among others. In many respects, knowledge repositories serve as the cybersecurity community "memory" and enable policy interoperability. Examples include Information Assurance Checklists housed on the National Checklist Program website (<http://checklists.nist.gov/>), Department of Defense Security Technical Implementation Guides (STIGs), and vendor guides."

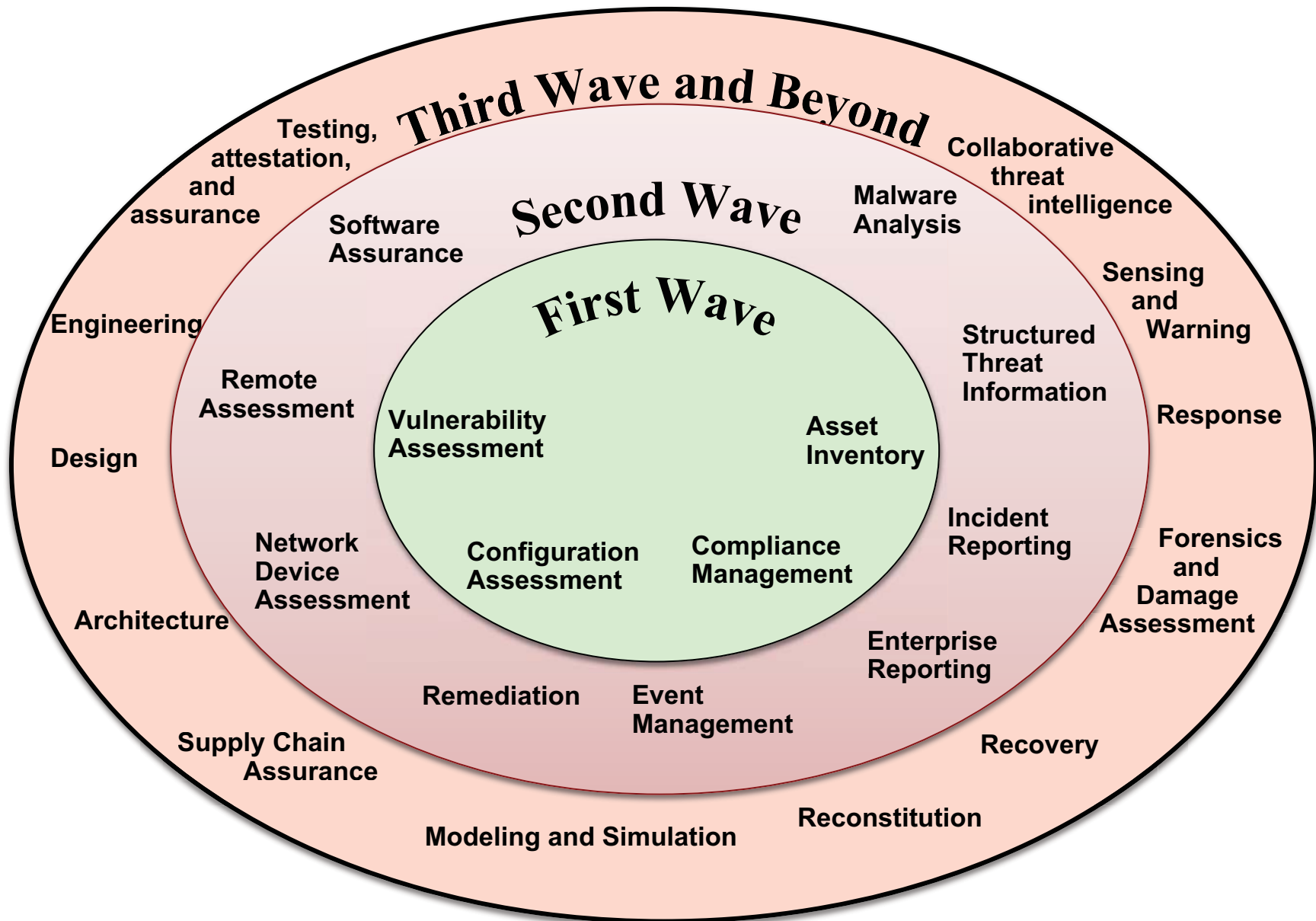
Figure 4 presents a history of U.S. Government supported security content automation efforts along with projected achievements through 2014. Projections are based on current resourcing and the interests of a largely volunteer and self-directed community. Figure 4 also illustrates how standards build upon themselves to expand functionality over time (e.g., the expansion of configuration management capabilities from desktops to networks).

¹³ See the Glossary at the end of this paper for the full name of the various named standards.

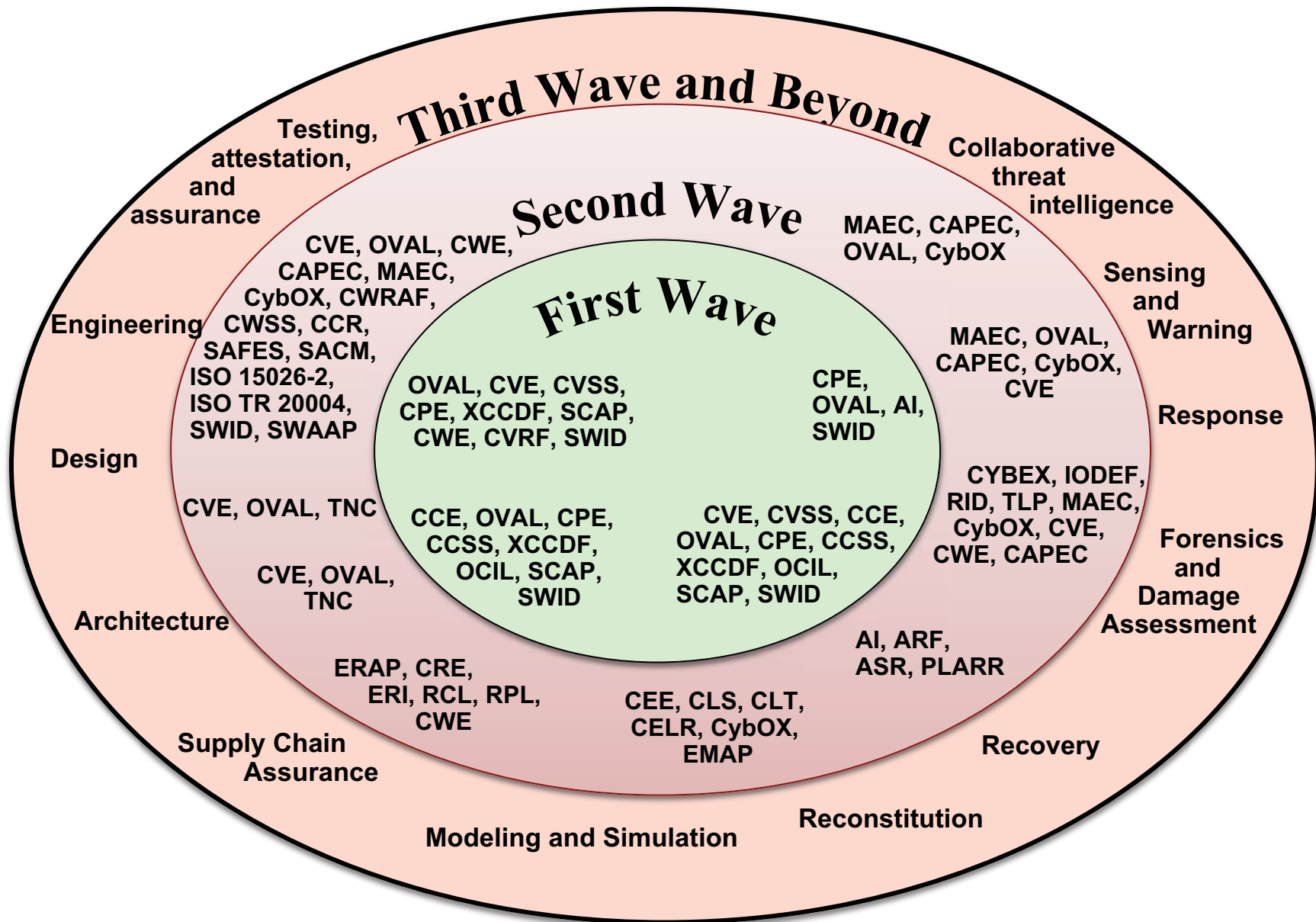
March 23, 2011

12

“Enabling Distributed Security in Cyberspace: Building a Healthy and Resilient Cyber Ecosystem with Automated Collective Action”



Ecosystem Areas Directly Enabled/Supported by Enumerations/Languages



[Click Here to Install Silverlight](#)

United States [Change](#) | [All Microsoft Sites](#)

Microsoft | TechNet

Search Microsoft.com

bing Web

[TechNet Home](#)

[TechCenters](#)

[Downloads](#)

[TechNet Program](#)

[Subscriptions](#)

[Security Bulletins](#)

[Archive](#)

Search for

Go

TechNet Security

Security Bulletin Search

Library

Learn

Downloads

Support

[TechNet Home](#) > [TechNet Security](#) > [Bulletins](#)

Microsoft Security Bulletin MS10-071 - Critical Cumulative Security Update for Internet Explorer (2360131)

Published: October 12, 2010 | Updated: October 13, 2010

Version: 1.1

General Information

Executive Summary

This security update resolves seven privately reported vulnerabilities and three publicly disclosed vulnerabilities in Internet Explorer. The most severe vulnerabilities could allow remote code execution if a user views a specially crafted Web page using Internet Explorer. Users whose accounts are configured to have fewer user rights on the system could be less impacted than users who operate with administrative user rights.

[↑ Top of section](#)

[Frequently Asked Questions \(FAQ\) Related to This Security Update](#)

Vulnerability Information

[Severity Ratings and Vulnerability Identifiers](#)

- [AutoComplete Information Disclosure Vulnerability - CVE-2010-0808](#)
- [HTML Sanitization Vulnerability - CVE-2010-3243](#)
- [HTML Sanitization Vulnerability - CVE-2010-3324](#)
- [CSS Special Character Information Disclosure Vulnerability - CVE-2010-3325](#)
- [Uninitialized Memory Corruption Vulnerability - CVE-2010-3326](#)
- [Anchor Element Information Disclosure Vulnerability - CVE-2010-3327](#)
- [Uninitialized Memory Corruption Vulnerability - CVE-2010-3328](#)
- [Uninitialized Memory Corruption Vulnerability - CVE-2010-3329](#)
- [Cross-Domain Information Disclosure Vulnerability - CVE-2010-3330](#)
- [Uninitialized Memory Corruption Vulnerability - CVE-2010-3331](#)



([Sign In/Register for Account](#) | [Help](#))

United States ▾

Communities ▾

I am a... ▾

I want to... ▾

Secure Search 🔍

Sun Quick Links ▾

Products and Services

Downloads

Store

Support

Education

Partners

About

Oracle Technology Network ▾

Oracle Technology Network > Topics > Security

Embedded

BI & Data Warehousing

.NET

Linux

PHP

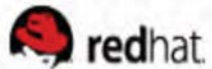
Oracle Critical Patch Update Advisory - October 2010

Description

A Critical Patch Update is a collection of patches for multiple security vulnerabilities. It also includes non-security fixes that are required (because of interdependencies) by those security patches. Critical Patch Updates are cumulative, except as noted below, but each advisory describes only the security fixes added since the previous Critical Patch Update. Thus, prior Critical Patch Update Advisories should be reviewed for information regarding earlier accumulated security fixes. Please refer to:

Oracle Database Server Risk Matrix

CVE#	Component	Protocol	Package and/or Privilege Required	Remote Exploit without Auth.?	CVSS VERSION 2.0 RISK (see Risk Matrix Definitions)							Last Affected Patch set (per Supported Release)	Notes
					Base Score	Access Vector	Access Complexity	Authentication	Confidentiality	Integrity	Availability		
CVE-2010-2390 (Oracle Enterprise Manager Grid Control)	EM Console	HTTP	None	Yes	7.5	Network	Low	None	Partial+	Partial+	Partial+	10.1.0.5, 10.2.0.3	See Note 1
CVE-2010-2419	Java Virtual Machine	Oracle Net	Create Session	No	6.5	Network	Low	Single	Partial+	Partial+	Partial+	10.1.0.5, 10.2.0.4, 11.1.0.7, 11.2.0.1	
CVE-2010-1321	Change Data Capture	Oracle Net	Execute on DBMS_CDC_PUBLISH	No	5.5	Network	Low	Single	Partial+	Partial+	None	-	See Note 2
CVE-2010-2412	OLAP	Oracle Net	Create Session	No	5.5	Network	Low	Single	Partial+	Partial+	None	11.1.0.7	
CVE-2010-2415	Change Data Capture	Oracle Net	Execute on DBMS_CDC_PUBLISH	No	4.9	Network	Medium	Single	Partial+	Partial+	None	10.1.0.5, 10.2.0.4, 11.1.0.7, 11.2.0.1	
CVE-2010-2411	Job Queue	Oracle Net	Execute on SYS.DBMS_IJOB	No	4.6	Network	High	Single	Partial+	Partial+	Partial+	-	See Note 2
CVE-2010-2407	SDK	HTTP	None	Yes	4.3	Network	Medium	None	None	Partial	None	10.1.0.5, 10.2.0.4, 11.1.0.7	
CVE-2010-2391	Core RDBMS	Oracle Net	Create Session	No	3.6	Network	High	Single	Partial	Partial	None	10.1.0.5, 10.2.0.3	
CVE-2010-2389 (Oracle Fusion Middleware)	Perl	Oracle Net	Local Logon	No	1.0	Local	High	Single	None	Partial+	None	-	See Note 2

[Errata](#)[Log In](#)[About RHN](#)

Important: kernel security and bug fix update

Advisory: [RHSA-2010:0723-1](#)

Type: Security Advisory

Severity: Important

Issued on: 2010-09-29

Last updated on: 2010-09-29

Affected Products: [Red Hat Enterprise Linux \(v. 5 server\)](#)
[Red Hat Enterprise Linux Desktop \(v. 5 client\)](#)

OVAL: [com.redhat.rhsa-20100723.xml](#)

CVEs (cve.mitre.org): [CVE-2010-1083](#)
[CVE-2010-2492](#)
[CVE-2010-2798](#)
[CVE-2010-2938](#)
[CVE-2010-2942](#)
[CVE-2010-2943](#)
[CVE-2010-3015](#)

Mailing Lists

Apple Mailing Lists



Search!

☐ Search only in security-announce list

[\[Date Prev\]](#)[\[Date Next\]](#)[\[Thread Prev\]](#)[\[Thread Next\]](#)[\[Date Index\]](#)[\[Thread Index\]](#)

APPLE-SA-2010-08-11-1 iOS 4.0.2 Update for iPhone and iPod touch

Subject: APPLE-SA-2010-08-11-1 iOS 4.0.2 Update for iPhone and iPod touch

From: Apple Product Security <email@hidden>

Date: Wed, 11 Aug 2010 12:19:43 -0700

Delivered-to: email@hidden

Delivered-to: email@hidden

-----BEGIN PGP SIGNED MESSAGE-----

Hash: SHA1

APPLE-SA-2010-08-11-1 iOS 4.0.2 Update for iPhone and iPod touch

iOS 4.0.2 Update for iPhone and iPod touch is now available and addresses the following:

FreeType

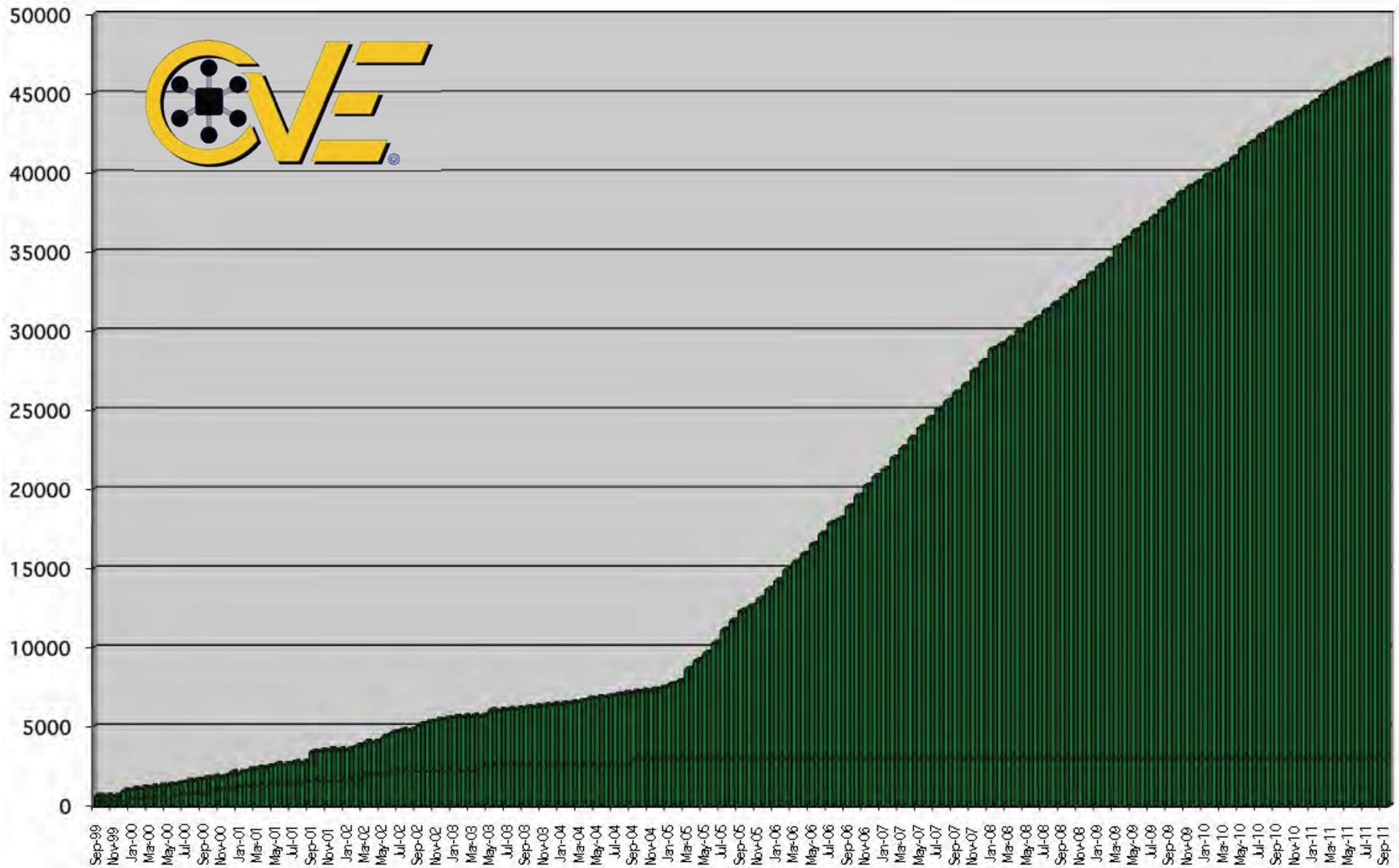
CVE-ID: CVE-2010-1797

Available for: iOS 2.0 through 4.0.1 for iPhone 3G and later, iOS 2.1 through 4.0 for iPod touch (2nd generation) and later

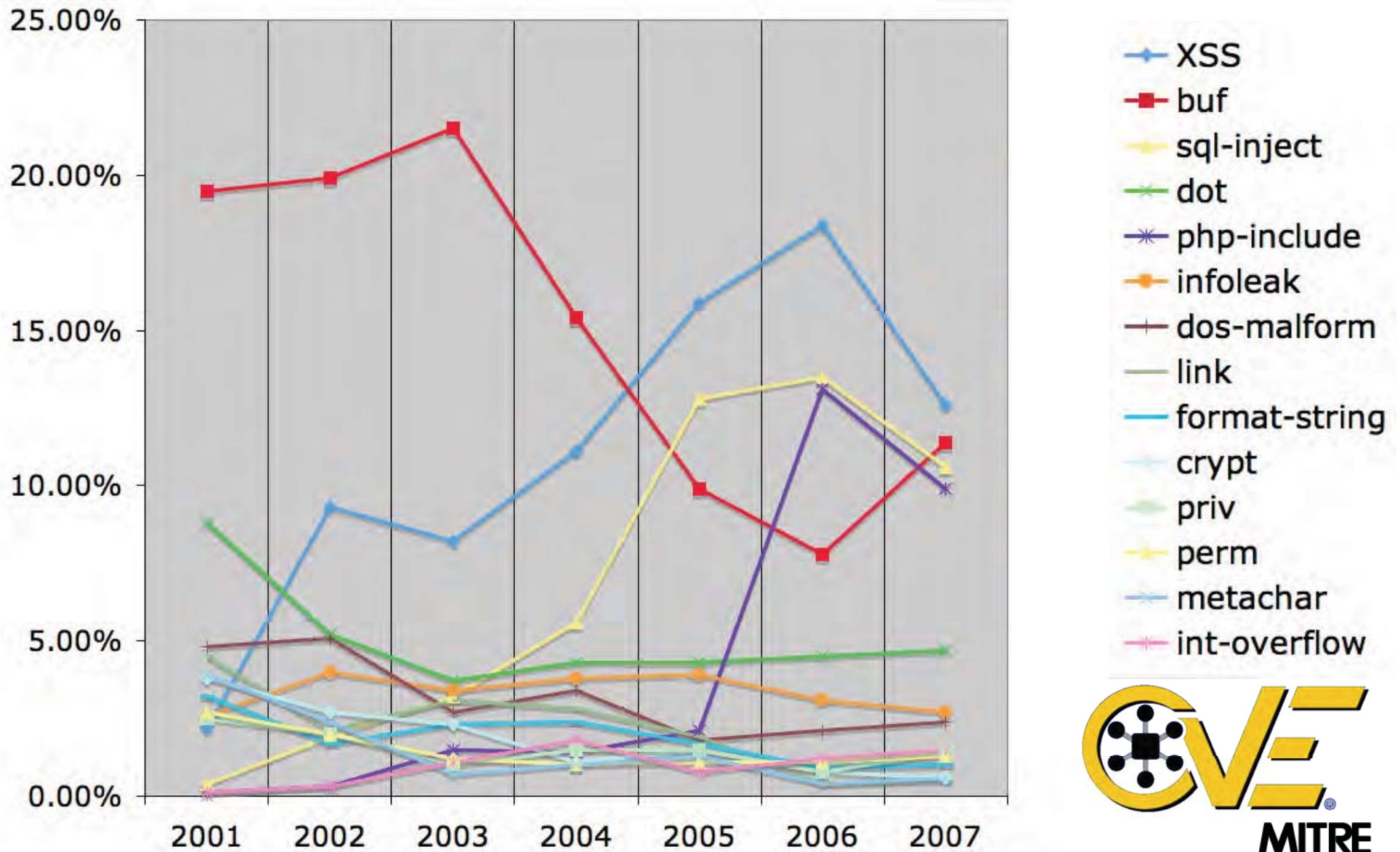
Impact: Viewing a PDF document with maliciously crafted embedded fonts may allow arbitrary code execution

Description: A stack buffer overflow exists in FreeType's handling of CFF encodes. Viewing a PDF document with maliciously crafted

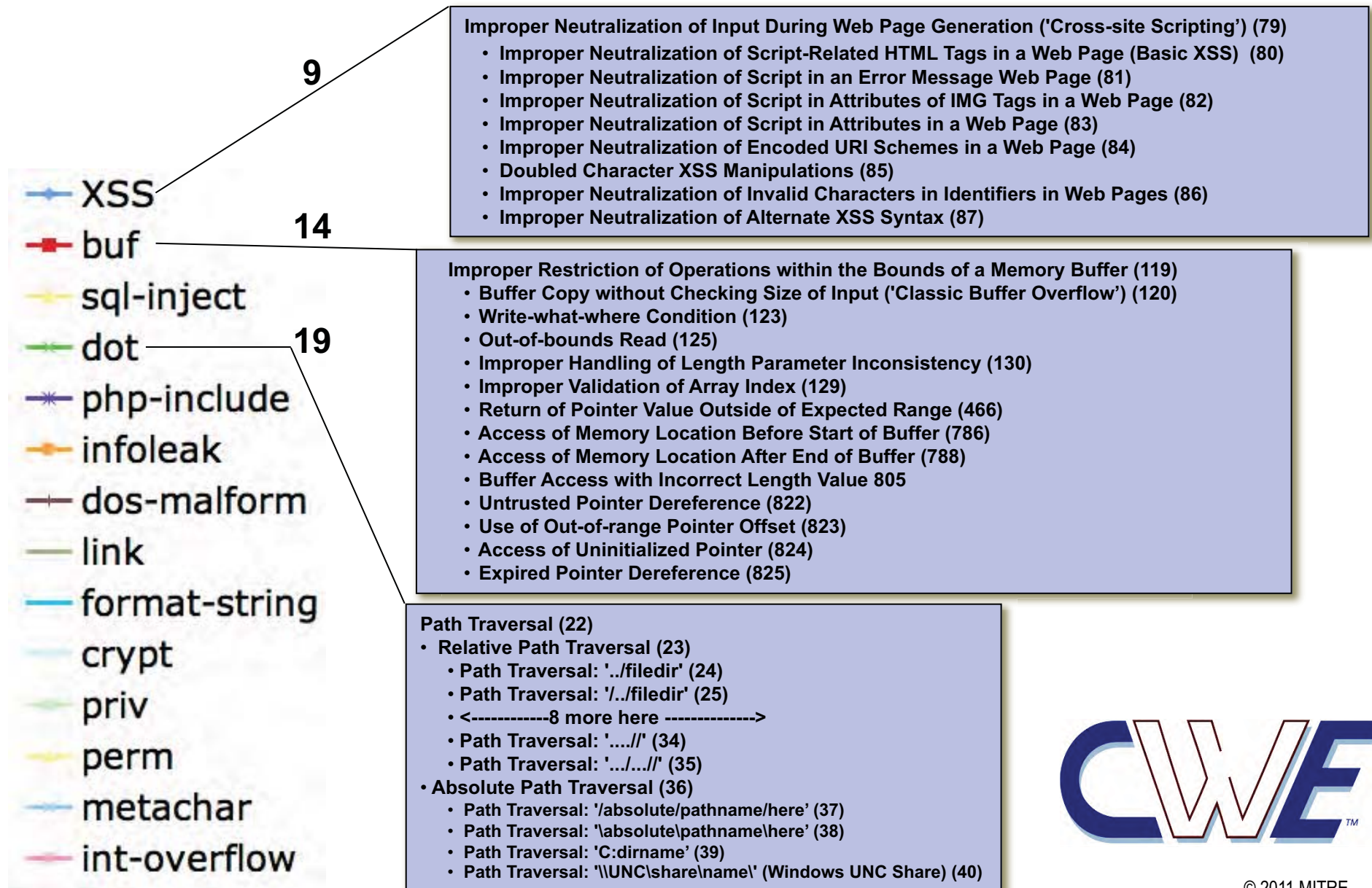
CVE 1999 to 2011



Vulnerability Type Trends: A Look at the CVE List (2001 - 2007)



Removing and Preventing the Vulnerabilities Requires More Specific Definitions...CWEs



**Wouldn't it be nice
if the weaknesses
in software were as
easy to spot and
their impact as
easy to understand
as a screen door in
a submarine...**



Linkage with Fundamental Changes in Enterprise Security Initiatives

Enabling Distributed Security in Cyberspace

Building a Healthy and Resilient Cyber Ecosystem with Automated Collective Action

March 23, 2011

- **Technical Interoperability.** The ability for different technologies to communicate and exchange data based upon well defined and widely adopted interface standards.
- **Policy Interoperability.** Common business processes related to the transmission, receipt, and acceptance of data among participants.

Within cybersecurity, all three types of interoperability are being enabled through an approach that has been refined over the past decade by many in industry, academia, and government. It is an information-oriented approach, generally referred to as [cyber] security content automation and comprises the following elements.¹³

- **Enumerations.** These are lists or catalogs of the fundamental entities of cybersecurity, for example, cyber devices and software items (CPE); device and software weaknesses in architecture, design, or code (CWE); or publicly known attack patterns (CAPEC).





The Security Development Lifecycle

Welcome to MSDN Blogs [Sign in](#) | [Join](#) | [Help](#)

SEARCH

[HOME](#)
[EMAIL](#)
[RSS 2.0](#)
[ATOM 1.0](#)

Recent Posts

MS08-078 and the SDL
Announcing CAT.NET CTP and AntixSS v3 beta
SDL videos
BlueHat SDL Sessions Wrap-up
Secure Coding Secrets?

Tags

Common Criteria [Crawl Walk Run](#)
Privacy [SDL](#) [SDL Pro Network](#)
Security Assurance [Security Blackhat](#)
SDL [threat modeling](#)

News

Blogroll

BlueHat Security Briefings
The Microsoft Security Response Center
Michael Howard's Web Log
The Data Privacy Imperative
Security Vulnerability Research & Defense
Visual Studio Code Analysis Blog
MSRC Ecosystem Strategy Team

Books / Papers / Guidance

The Security Development Lifecycle (Howard and Lipner)
Privacy Guidelines for Developing Software Products and Services
Microsoft Security Development Lifecycle (SDL) - Portal
Microsoft Security Development Lifecycle (SDL) - Process Guidance (Web)
Microsoft Security Development Lifecycle (SDL) - Process Guidance (.doc)

MS08-078 and the SDL ★★★★★

Hi, Michael here.

Every bug is an opportunity to learn, and the security update that fixed the data binding bug that affected Internet Explorer users is no exception.

The Common Vulnerabilities and Exposures (CVE) entry for this bug is [CVE-2008-4844](#).

Before I get started, I want to explain the goals of the SDL and the security work here at Microsoft. The SDL is designed as a multi-layered process to help systemically reduce security vulnerabilities; if one component of the SDL process fails to prevent or catch a bug, then some other component should prevent or catch the bug. The SDL also mandates the use of security defenses whose impact will be reflected in the "mitigations" section of a security bulletin, because we know that no software development process will catch all security bugs. As we have said many times, the goal of the SDL is to "Reduce vulnerabilities, and reduce the severity of what's missed."

In this post, I want to focus on the SDL-required code analysis, code review, fuzzing and compiler and operating system defenses and how they fared.

Background

The bug was an invalid pointer dereference in MSHHTML.DLL when the code handles data binding. It's important to point out that there is no heap corruption and there is no heap-based buffer overrun!

When data binding is used, IE creates an object which contains an array of data binding objects. In the code in question, when a data binding object is released, the array length is not correctly updated leading to a function call into freed memory.

The vulnerable code looks a little like this (by the way, the real array name is `_aryPXfer`, but I figured `ArrayOfObjectsFromIE` is a little more descriptive for people not in the Internet Explorer team.)

```
int MaxIdx = ArrayOfObjectsFromIE.Size()-1;
for (int i=0; i <= MaxIdx; i++) {
    if (!ArrayOfObjectsFromIE[i])
        continue;
    ArrayOfObjectsFromIE[i]->TransferFromSource();
    ...
}
```

Here's how the vulnerability manifests itself: if there are two data transfers with the same identifier (so `MaxIdx` is 2), and the first transfer updates the length of the `ArrayOfObjectsFromIE` array when its work was done and releases its data binding object, the loop count would still be whatever `MaxIdx` was at the start of the loop, 2.

This is a time-of-check-time-of-use (TOCTOU) bug that led to code calling into a freed memory block. The Common Weakness Enumeration (CWE) classification for this vulnerability is [CWE-367](#).

The fix was to check the maximum iteration count on each loop iteration rather than once before the loop starts. This is the correct fix for a TOCTOU bug - move the check as close as possible to the action because in

a time-of-check-time-of-use (TOCTOU) bug that led to code calling into a freed memory block. The on Weakness Enumeration (CWE) classification for this vulnerability is [CWE-367](#).

September 2008 (5)
August 2008 (2)
July 2008 (8)
June 2008 (4)

TOCTOU issues. We will update our training to address this.

Our static analysis tools don't find this because the tools would need to understand the re-entrant nature of the code.

Fuzz Testing

CWE List

Full Dictionary View
Development View
Research View
Reports

About

Sources
Process
Documents

Community

Related Activities
Discussion List
Research
CWE/SANS Top 25
CWSS

News

Calendar
Free Newsletter

Compatibility

Program
Requirements
Declarations
Make a Declaration

Contact Us

Search the Site

CWE-367: Time-of-check Time-of-use (TOCTOU) Race Condition

Time-of-check Time-of-use (TOCTOU) Race Condition

Weakness ID: 367 (Weakness Base)

Status: Incomplete

Description

Description Summary

The software checks the state of a resource before using that resource, but the resource's state can change between the check and the use in a way that invalidates the results of the check. This can cause the software to perform invalid actions when the resource is in an unexpected state.

Extended Description

This weakness can be security-relevant when an attacker can influence the state of the resource between check and use. This can happen with shared resources such as files, memory, or even variables in multithreaded programs.

Alternate Terms

TOCTTOU: The TOCTTOU acronym expands to "Time Of Check To Time Of Use". Usage varies between TOCTOU and TOCTTOU.

Time of Introduction

- Implementation

Applicable Platforms

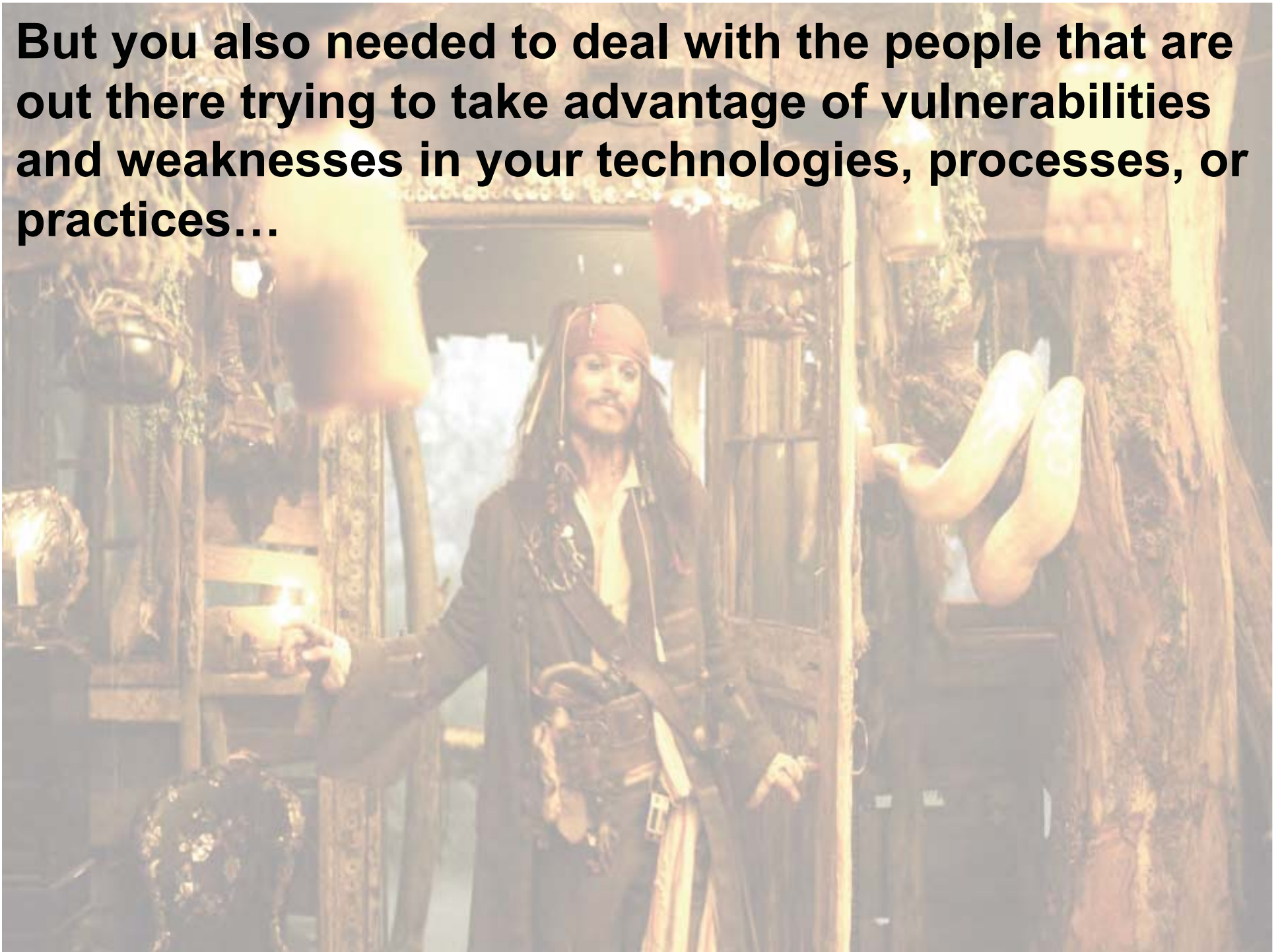
Languages

All

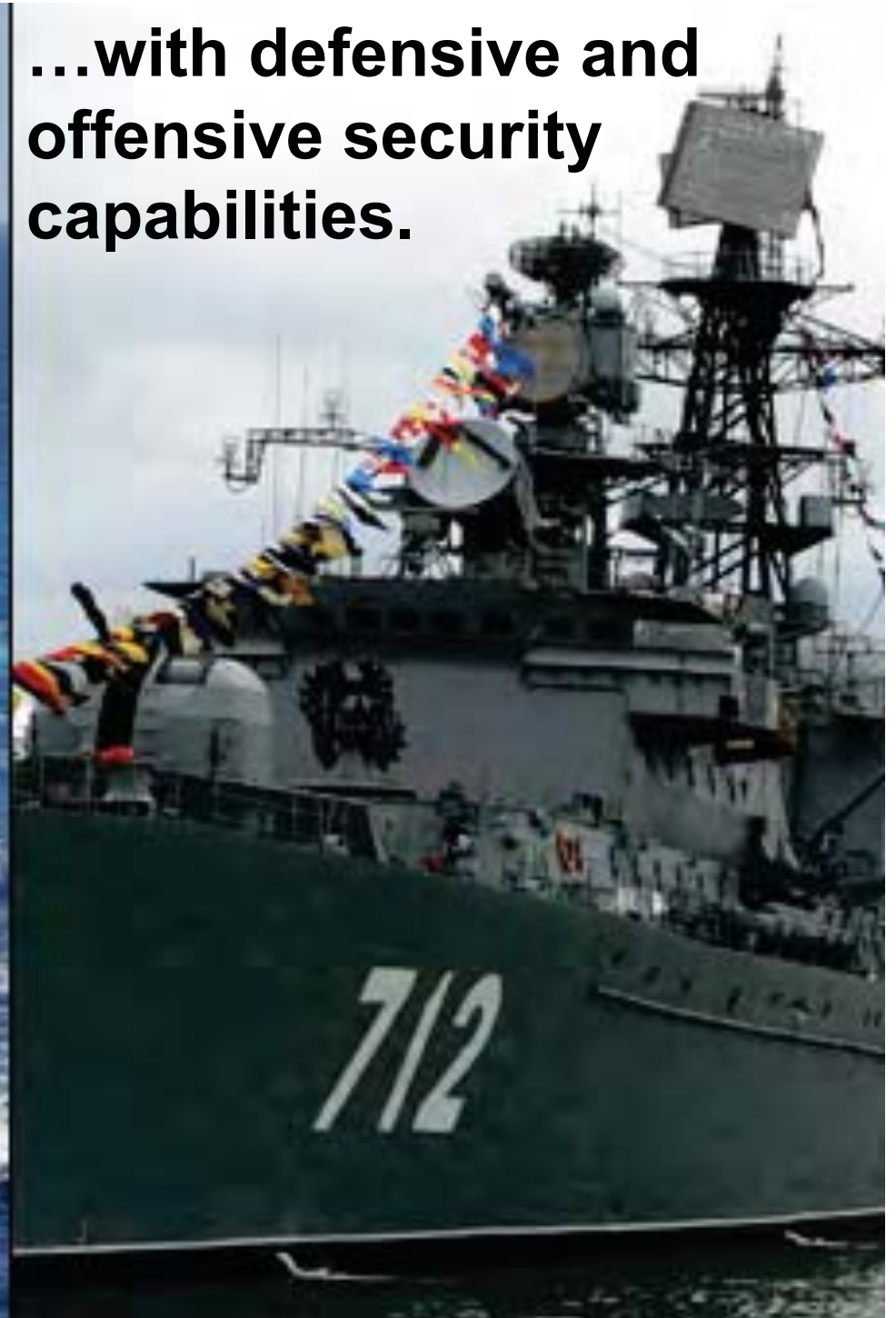
Common Consequences

Scope	Effect
Access Control	The attacker can gain access to otherwise unauthorized resources.
Access Control Authorization	Race conditions such as this kind may be employed to gain read or write access to resources which are not normally readable or writable by the user in question.
Integrity	The resource in question, or other resources (through the corrupted one), may be changed in undesirable ways by a malicious user.
Accountability	If a file or other resource is written in this method, as opposed to in a valid way, logging of the activity may not occur.
Non-Repudiation	In some cases it may be possible to delete files a malicious user might not otherwise have access to, such as log files.

But you also needed to deal with the people that are out there trying to take advantage of vulnerabilities and weaknesses in your technologies, processes, or practices...



**...with defensive and
offensive security
capabilities.**





XSS (CWE-79)
Exploit
(CAPEC-86)

**Security
Feature**

SQL Injection
(CWE-89)
Exploit
(CAPEC-66)

CWE List

Full Dictionary View
Development View
Research View
Reports

About

Sources
Process
Documents

Community

Related Activities
Discussion List
Research
CWE/SANS Top 25
CWSS

News

Calendar
Free Newsletter

Compatibility

Program
Requirements
Declarations
Make a Declaration

Contact Us

Search the Site

CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

Weakness ID: 89 (*Weakness Base*)

Status: Draft

Description

Description Summary

The software constructs all or part of an SQL command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended SQL command when it is sent to a downstream component.

Extended Description

Without sufficient removal or quoting of SQL syntax in user-controllable inputs, the generated SQL query can cause those inputs to be interpreted as SQL instead of ordinary user data. This can be used to alter query logic to bypass security checks, or to insert additional statements that modify the back-end database, possibly including execution of system commands.

SQL injection has become a common issue with database-driven web sites. The flaw is easily detected, and easily exploited, and as such, any site or software package with even a minimal user base is likely to be subject to an attempted attack of this kind. This flaw depends on the fact that SQL makes no real distinction between the control and data planes.

Time of Introduction

- Architecture and Design
- Implementation
- Operation

Applicable Platforms

Languages

All

Technology Classes

Database-Server

Rank	Score	ID	Name
[1]	93.8	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
[2]	83.3	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
[3]	79.0	CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
[4]	77.7	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
[5]	76.9	CWE-306	Missing Authentication for Critical Function
[6]	76.8	CWE-862	Missing Authorization
[7]	75.0	CWE-798	Use of Hard-coded Credentials
[8]	75.0	CWE-311	Missing Encryption of Sensitive Data
[9]	74.0	CWE-434	Unrestricted Upload of File with Dangerous Type
[10]	73.8	CWE-807	Reliance on Untrusted Inputs in a Security Decision
[11]	73.1	CWE-250	Execution with Unnecessary Privileges
[12]	70.1	CWE-352	Cross-Site Request Forgery (CSRF)
[13]	69.3	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
[14]	68.5	CWE-494	Download of Code Without Integrity Check
[15]	67.8	CWE-863	Incorrect Authorization

up from 2	+1
up from 9	+7
same	0
down from 1	-3
up from 19	+14
split of prior #5	-1
up from 11	+4
up from 10	+2
down from 8	-1
down from 6	-4
new entry	n/a
down from 4	-8
down from 7	-6
up from 20	+6
split of prior #5	-10



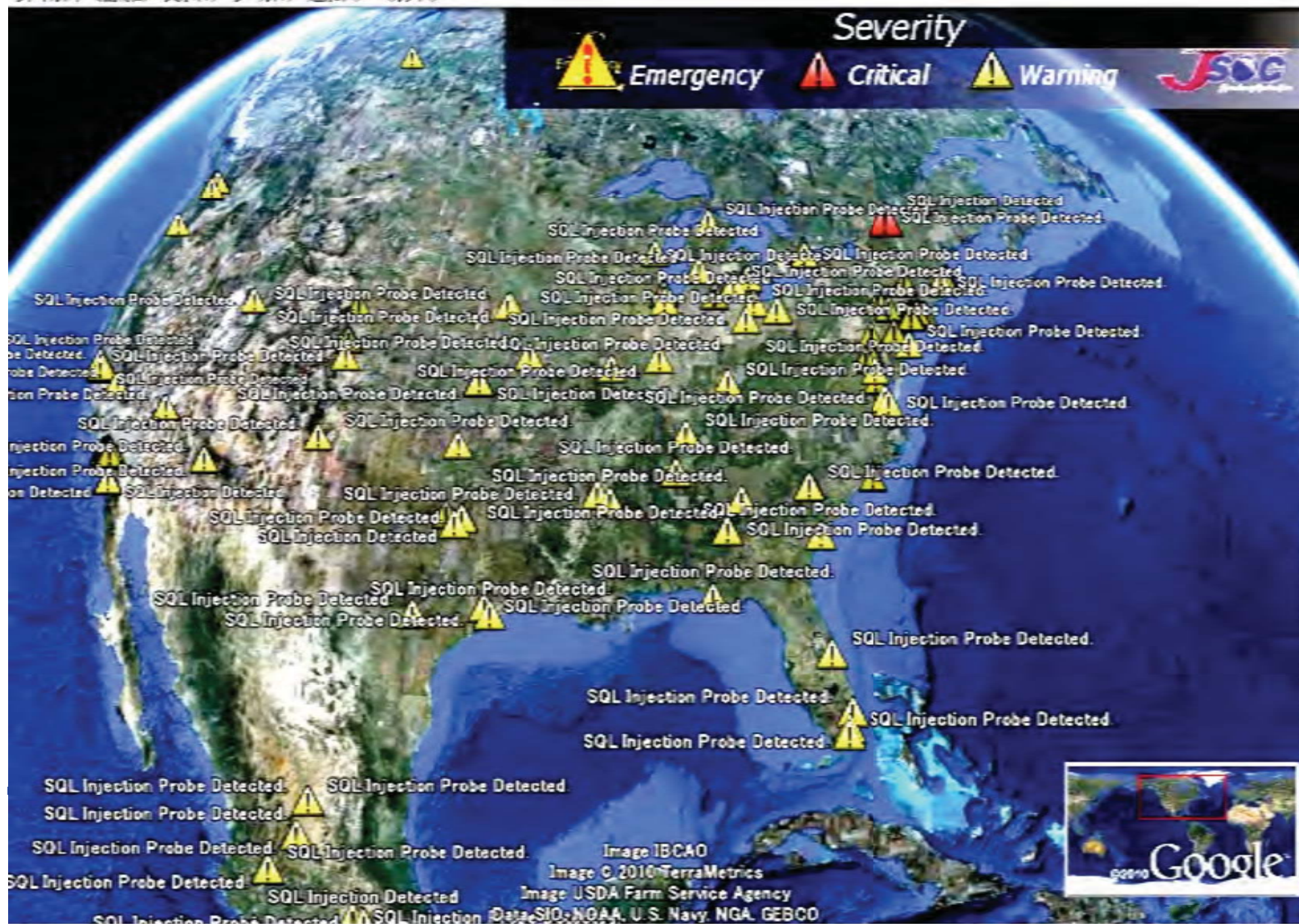
Emergency



Critical



Warning



CWE web site visitors by City



CWE web site visitors by Country



IBM Software Technical White Paper

One way to improve software security is to gain a better understanding of the most common weaknesses that can affect software security. With that in mind, there are many resources available online to help organizations learn about

Resources available to help organizations protect systems in

Resource	Focus
DoD Information Assurance Certification and Accreditation Process (DIACAP)	The DIACAP defines the minimum standards accredited by the DoD and authorized to application-level security controls, but it is activities, general tasks, and a management
Defense Information Systems Agency (DISA)	The DISA provides a security technical in development that offer more granular information on application- or software-level control ability assessment techniques. The checklist is the same one used by DoD auditors.
U.S. Department of Homeland Security (DHS)	The DHS offers information on security best practices and tools for application- and software part of its "Build Security In" initiative.
The Common Weaknesses Enumeration project, a community-based program sponsored by the MITRE Corporation, an IBM Business Partner	The MITRE Corporation maintains the online common vulnerabilities and exposures (CVE) enumeration (CWE) knowledge bases about currently known vulnerabilities and types of knowledge base focuses on packaged software and deals with patches and known vulnerabilities. knowledge base focuses on code vulnerabilities.
The Open Web Application Security Project (OWASP)	One of the best sources for information on web application security issues, the OWASP 10 list of the most dangerous and most commonly found and commonly exploited vulnerabilities how to identify, fix and avoid them.
Digital Building Security In Maturity Model (BSIMM)	Created by Digital, an IBM Business Partner, the BSIMM is designed to help organizations and plan a software security initiative. The focus is on making applications more secure, process and at later stages in the software life cycle.
IBM X-Force™ research and development team	A global cyberthreat and risk analysis team that monitors traffic and attacks around the world. IBM X-Force team is an excellent resource for trend analysis and answers to questions about attacks are most common, where they are coming from and what organizations can do to the risks.
IBM Institute for Advanced Security (IAS)	This companywide cybersecurity initiative applies IBM research, services, software and tools to help governments and other clients improve the security and resiliency of their IT and business

Test and vulnerability assessment

Testing applications for security defects should be an integral and organic part of any software testing process. During security testing, organizations should test to help ensure that the security requirements have been implemented and the product is free of vulnerabilities.

The SEF refers to the MITRE Common Weakness Enumeration⁵ (CWE) list and the Common Vulnerability Enumeration⁶ (CVE) list. This information and vulnerability assessment against the m

Creating a security plan includes

⁵ For more information
⁶ For more information

10 Security in Development:

Security in Development: The IBM Secure Engineering Framework



Redguides
for Business Leaders

Danny Allan
Tim Hahn
Andras Szakal
Jim Whitmore
Axel Buecker

- Investigating common development processes and the IBM Integrated Product Development process
- Emphasizing security awareness and requirements in the software development process
- Discussing test and vulnerability assessments



Redbooks

Making the Business Case for Software Assurance

Nancy R. Mead
Julia H. Allen
W. Arthur Conklin
Antonio Drommi
John Harrison
Jeff Ingalsbe
James Rainey
Dan Shoemaker

April 2009

SPECIAL REPORT
CMU/SEI-2009-SR-001

CERT Program
Unlimited distribution subject to the copyright.

<http://www.sei.cmu.edu>



Carnegie Mellon

OVM: An Ontology for Vulnerability Management

Ju An Wang & Minzhe Guo
Southern Polytechnic State University
1100 South Marietta Parkway
Marietta, GA 30060
(01) 678-915-3718

jwang@spsu.edu

ABSTRACT

In order to reach the goals of the Information Security Automation Program (ISAP) [1], we propose an ontological approach to capturing and utilizing the fundamental concepts in information security and their relationship, retrieving vulnerability data and reasoning about the cause and impact of vulnerabilities. Our ontology for vulnerability management (OVM) has been populated with all vulnerabilities in NVD [2] with additional inference rules, knowledge representation, and data-mining mechanisms. With the seamless integration of common vulnerabilities and their related concepts such as attacks and countermeasures, OVM provides a promising pathway to making ISAP successful.

Categories and Subject Descriptors

C.2.0 [Computer-Communication Networks]: General [Security and protection]; K.6.5 [Management of Computing and Information Systems]: Security and Protection;

General Terms

Ontology, Security, Vulnerability Analysis and Management

Keywords

Security vulnerability, Semantic technology, Ontology, Vulnerability analysis

1. INTRODUCTION

The Information Security Automation Program (ISAP) is a U.S. government multi-agency initiative to enable automation and standardization of technical security operations [1]. Its high-level goals include standards based automation of security checking and remediation as well as automation of technical compliance activities. Its low-level objectives include enabling standards based communication of vulnerability data, customizing and managing configuration baselines for various IT products, assessing information systems and reporting compliance status, using standard metrics to weight and aggregate potential vulnerability impact, and remediating identified vulnerabilities [1]. Secure computer systems ensure that confidentiality, integrity, and availability are maintained for users, data, and other information assets. Over the past a few decades, a significantly large amount of knowledge has been accumulated in the area of information security. However, a lot of concepts in information security are vaguely defined and sometimes they have different

semantics in different contexts, causing misunderstanding among stake holders due to the language ambiguity. On the other hand, the standardization, design and development of security tools [1-5] require a systematic classification and definition of security concepts and techniques. It is important to have a clearly defined vocabulary and standardized language as means to accurately communicate system vulnerability information and their countermeasures among all the people involved. We believe that semantic technology in general, and ontology in particular, could be a useful tool for system security. Our research work has confirmed this belief and this paper will report some of our work in this area.

An ontology is a specification of concepts and their relationship. Ontology represents knowledge in a formal and structured form. Therefore, ontology provides a better tool for communication, reusability, and organization of knowledge. Ontology is a knowledge representation (KR) system based on Description Logics (DLs) [6], which is an umbrella name for a family of KR formalisms representing knowledge in various domains. The DL formalism specifies a knowledge domain as the "world" by first defining the relevant concepts of the domain, and then it uses these concepts to specify properties of objects and individuals occurring in the domain [10-12]. Semantic technologies not only provide a tool for communication, but also a foundation for high-level reasoning and decision-making. Ontology, in particular, provides the potential of formal logic inference based on well-defined data and knowledge bases. Ontology captures the relationships between collected data and use the explicit knowledge of concepts and relationships to deduce the implicit and inherent knowledge. As a matter of fact, a heavy-weight ontology could be defined as a formal logic system, as it includes facts and rules, concepts, concept taxonomies, relationships, properties, axioms and constraints.

A vulnerability is a security flaw, which arises from computer system design, implementation, maintenance, and operation. Research in the area of vulnerability analysis focuses on discovery of previously unknown vulnerabilities and quantification of the security of systems according to some metrics. Researchers at MITRE have provided a standard format for naming a security vulnerability, called Common Vulnerabilities and Exposures (CVE) [14], which assigns each vulnerability a unique identification number. We have designed a vulnerability ontology OVM (ontology for vulnerability management) populated with all existing vulnerabilities in NVD [2]. It supports research on reasoning about vulnerabilities and characterization of vulnerabilities and their impact on computing systems. Vendors and users can use our ontology in support of vulnerability analysis, tool development and vulnerability management.

The rest of this paper is organized as follows: Section 2 presents the architecture of our OVM. Section 3 discusses how to populate the OVM with vulnerability instances from NVD and other

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CSIIRW '09, April 13-15, Oak Ridge, Tennessee, USA
Copyright © 2009 ACM 978-1-60558-518-5 ... \$5.00

16 July 2010

A Human Capital Crisis in Cybersecurity

Technical Proficiency Matters

A White Paper of the
CSIS Commission on Cybersecurity for the 44th Presidency

COCHAIRS
Representative James R. Langevin
Representative Michael T. McCaul
Scott Charney
Lt. General Harry Raduege,
USAF (ret.)

PROJECT DIRECTOR
J...

based on a body of knowledge that represents the complete set of concepts, terms and activities that make up a professional domain. And absent such a body of knowledge there is little basis for supporting a certification program. Indeed it would be dangerous and misleading.

A complete body of knowledge covering the entire field of software engineering may be years away. However, the body of knowledge needed by professionals to create software free of common and critical security flaws has been developed, vetted widely and kept up to date. That is the foundation for a certification program in software assurance that can gain wide adoption. It was created in late 2008 by a consortium of national experts, sponsored by DHS and NSA, and was updated in late 2009. It contains ranked lists of the most common errors, explanations of why the errors are dangerous, examples of those errors in multiple languages, and ways of eliminating those errors. It can be found at <http://cwe.mitre.org/top25>.

Any programmer who writes code without being aware of those problems and is not capable of writing code free of those errors is a threat to his or her employers and to others who use computers connected to systems running his or her software.

A complete body of knowledge covering the entire field of software engineering may be years away. However, the body of knowledge needed by professionals to create software free of common and critical security flaws has been developed, vetted widely and kept up to date. That is the foundation for a certification program in software assurance that can gain wide adoption. It was created in late 2008 by a consortium of national experts, sponsored by DHS and NSA, and was updated in late 2009. It contains ranked lists of the most common errors, explanations of why the errors are dangerous, examples of those errors in multiple languages, and ways of eliminating those errors. It can be found at <http://cwe.mitre.org/top25>.

Any programmer who writes code without being aware of those problems and is not capable of writing code free of those errors is a threat to his or her employers and to others who use computers connected to systems running his or her software.



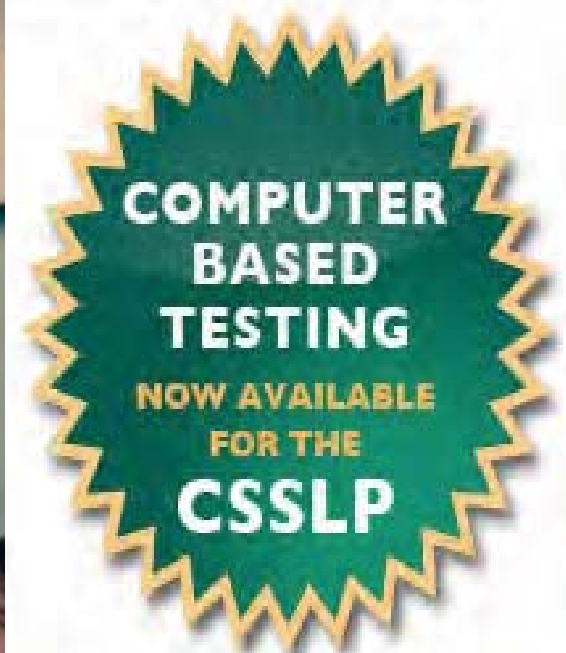
The **Certified Secure Software Lifecycle Professional (CSSLP)** Certification Program will show software lifecycle stakeholders not only how to implement security, but how to glean security requirements, design, architect, test and deploy secure software.

An Overview of the Steps:

(ISC)²® 5-day CSSLP CBK® Education Program

Educate yourself and learn security best practices and industry standards for the software lifecycle through the CSSLP Education Program. (ISC)² provides education your way to fit your life and schedule. Completing this course will, not only teach all of the

establish a security plan across your



Industry Uptake

Foreword

In 2008, the Software Assurance Forum for Excellence in Code (SAFECode) published the first version of this report in an effort to help others in the industry initiate or improve their own software assurance programs and encourage the industry-wide adoption of what we believe to be the most fundamental secure development methods. This work remains our most in-demand paper and has been downloaded more than 50,000 times since its original release.

However, secure software development is not only a goal, it is also a process. In the nearly two and a half years since we first released this paper, the process of building secure software has continued to evolve and improve alongside innovations and advancements in the information and communications technology industry. Much has been learned not only through increased community collaboration, but also through the ongoing internal efforts of SAFECode's member companies. This 2nd Edition aims to help disseminate that new knowledge.

Just as with the original paper, this paper is not meant to be a comprehensive guide to all possible secure development practices. Rather, it is meant to provide a foundational set of secure development practices that have been effective in improving software security in real-world implementations by SAFECode members across their diverse development environments.

It is important to note that these are the "practiced practices" employed by SAFECode members, which we identified through an ongoing analysis of our members' individual software security efforts. By

bringing these methods together and sharing them with the larger community, SAFECode hopes to move the industry beyond defining theoretical best practices to describing sets of software engineering practices that have been shown to improve the security of software and are currently in use at leading software companies. Using this approach, enables SAFECode to encourage the adoption of best practices that are proven to be effective and implementable even when requirements and development timelines are taken into account.

Though expanded, our key goals remain—keep it concise, actionable, and relevant.

What's New

This edition of the paper prescribes updated security practices that are relevant to the Design, Programming, and Testing phases of the software development process. Practices have been shown to be effective in diverse development environments, originally also covered Training, Requirements, and Documentation, though given detailed treatment in SAFECode's separate papers on security engineering training and software integrity in the global supply chain, and thus we have refined our focus in this paper to concentrate on the core areas of design, development and testing.

The paper also contains two important, additional sections for each listed practice that will further increase its value to implementers—Common Weakness Enumeration (CWE) references and Verification guidance.

The paper also contains two important, additional sections for each listed practice that will further increase its value to implementers—Common Weakness Enumeration (CWE) references and Verification guidance.



example,

are available that support the Threat Model process with automated analysis of designs and questions for possible mitigations, issue-tracking, and communication related to the process. Some practitioners have hoped their Threat Model process to the point where tools are used to automate as much of it as possible, raising the stability of the process and providing another layer of support with standard diagramming, integration with a threat database and cases, and execution of recurring tasks.

Verification plan is a derivative of the results of the Threat Model activity. The Threat Model itself will serve as a clear road map for verification, containing enough information that each threat and mitigation can be verified.

During verification, the Threat Model and mitigated threats, as well as the annotated structural diagrams, should also be made available to testers in order to help define further tests and refine the verification process. A revised Threat Model and verification results should be made an integral part of the activities required to declare code complete.

CWE References

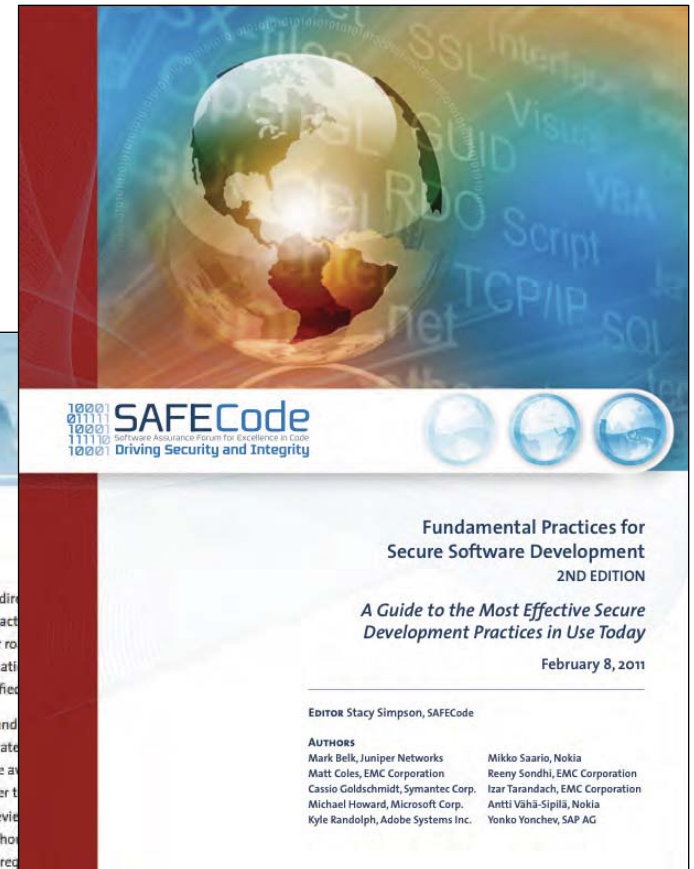
Much of CWE focuses on implementation issues, and Threat Modeling is a design-time event. There are, however, a number of CWEs that are applicable to the threat modeling process, including:

- CWE-287: Improper authentication is an example of weakness that could be exploited by a Spoofing threat
- CWE-264: Permissions, Privileges, and Access Controls is a parent weakness of many Tampering, Repudiation and Elevation of Privilege threats
- CWE-311: Missing Encryption of Sensitive Data is an example of an Information Disclosure threat
- CWE-400: (uncontrolled resource consumption) is one example of an unmitigated Denial of Service threat

An example of a portion of a test plan derived from a Threat Model could be:

Threat Identified	Design Element(s)	Mitigation	Verification
Session Hijacking	GUI	Ensure random session identifiers of appropriate length	Collect session identifiers over a number of sessions and examine distribution and length
Tampering with data in transit	Process A on server to Process B on client	Use SSL to ensure that data isn't modified in transit	Assert that communication cannot be established without the use of SSL

CWE





OWASP

The Open Web Application Security Project

Navigation

- ▶ Home
- ▶ News
- ▶ OWASP Projects
- ▶ Downloads
- ▶ Local Chapters
- ▶ Global Committees
- ▶ AppSec Job Board
- ▶ AppSec Conferences
- ▶ Presentations
- ▶ Video
- ▶ Press
- ▶ Get OWASP Books
- ▶ Get OWASP Gear
- ▶ Mailing Lists
- ▶ About OWASP
- ▶ Membership

Reference

- ▶ How To...
- ▶ Principles
- ▶ Threat Agents
- ▶ Attacks
- ▶ Vulnerabilities
- ▶ Controls
- ▶ Activities
- ▶ Technologies
- ▶ Glossary
- ▶ Code Snippets
- ▶ .NET Project
- ▶ Java Project

Language

- ▶ English
- ▶ Español

Code Review Introduction

««Code Review Guide History»»

Main
(Table of Contents)

»»Preparation»»

Contents [hide]

- 1 Introduction
 - 1.1 Why Does Code Have Vulnerabilities?
 - 1.2 What is Security Code Review?

Introduction

Code review is probably the single-most effective technique for identifying security flaws. When used together with automated tools and manual penetration testing, code review can significantly increase the cost effectiveness of an application security verification effort.

This guide does not prescribe a process for performing a security code review. Rather, this guide focuses on the mechanics of reviewing code for certain vulnerabilities, and provides limited guidance on how the effort should be structured and executed. OWASP intends to develop a more detailed process in a future version of this guide.

Manual security code review provides insight into the "real risk" associated with insecure code. This is the single most important value from a manual approach. A human reviewer can understand the context for certain coding practices, and make a serious risk estimate that accounts for both the likelihood of attack and the business impact of a breach.

Why Does Code Have Vulnerabilities?

MITRE has catalogued almost 700 different kinds of software weaknesses in their CWE project. These are all different ways that software developers can make mistakes that lead to insecurity. Every one of these weaknesses is subtle and many are seriously tricky. Software developers are not taught about these weaknesses in school and most do not receive any training on the job about these problems.

These problems have become so important in recent years because we continue to increase connectivity and to add technologies and protocols at a shocking rate. Our ability to invent technology has seriously outstripped our ability to secure it. Many of the technologies in use today simply have not received any security scrutiny.

There are many reasons why businesses are not spending the appropriate amount of time on security. Ultimately, these reasons stem from an underlying problem in the software market. Because software is essentially a black-box, it is extremely difficult to tell the difference between good code and insecure code. Without this visibility, buyers won't pay more for secure code, and vendors would be foolish to spend extra effort to produce secure code.

One goal for this project is to help software buyers gain visibility into the security of software and start to effect change in the software market.

Nevertheless, we still frequently get pushback when we advocate for security code review. Here are some of the (unjustified) excuses that we hear for not putting more effort into security:


"We never get hacked (that I know of), we don't need security"

Threat Classification Taxonomy Cross Reference View

Page history

✔ Check for plagiarism

Threat Classification 'Taxonomy Cross Reference View'

SideBar 

WASC ID	Name	CWE ID	CAPEC ID	SANS/CWE Top 25 2009	OWASP Top Ten 2010	OWASP Top Ten 2007	OWASP Top Ten 2004
WASC-01	Insufficient Authentication	287		642	A3 – Broken Authentication and Session Management, A4 – Insecure Direct Object References	A7 – Broken Authentication and Session Management, A4 – Insecure Direct Object Reference	A3 – Broken Authentication and Session management, A2 – Broken Access Control
WASC-02	Insufficient Authorization	284		285	A4 – Insecure Direct Object References, A7 – Failure to Restrict URL Access	A10 – Failure to Restrict URL Access, A4 – Insecure Direct Object Reference	A2 – Broken Access Control
WASC-03	Integer Overflows	190	128	682			
WASC-04	Insufficient Transport Layer Protection	311 523		319	A10 – Insufficient Transport Layer Protection	A9 – Insecure Communications	
WASC-05	Remote File Inclusion	98	193 253	426		A3 – Malicious File Execution	
WASC-06	Format String	134	67				
WASC-07	Buffer Overflow	119 120	10 100	119			A5 – Buffer Overflows
WASC-08	Cross-site Scripting	79	18 19 63	79	A2 – Cross-Site Scripting	A1 – Cross Site Scripting (XSS)	A4 – Cross Site Scripting (XSS)
WASC-09	Cross-site Request Forgery	352	62	352	A5 – Cross-Site Request Forgery	A5 – Cross Site Request Forgery (CSRF)	
WASC-10	Denial of Service	400	110	404	A7 – Failure to Restrict	A10 – Failure to	A0 – Denial of

- [Distributed Open Proxy Honeypots](#)
- [Script Mapping](#)
- [The Web Security Glossary](#)
- [Web Application Firewall Evaluation Criteria](#)
- [Web Application Security Scanner Evaluation Criteria](#)
- [Web Application Security Statistics](#)
- [Web Hacking Incidents Database](#)
- [WASC Threat Classification](#)

- [Robert Auger](#)
- [Ryan Barnett](#)
- [Romain Gaucher](#)
- [Sergey Gordeychik](#)
- [Ofer Shezaf](#)
- [Brian Shura](#)

- <http://www.webappsec.org/>

- <http://lists.webappsec.org/>

- <http://twitter.com/wascupdates>

- <http://www.linkedin.com/groups?qid=83336>

 **Insufficient Data Protection Working**
edited by Robert Auger

ISO/IEC JTC 1/SC 27/WG 3, NWP

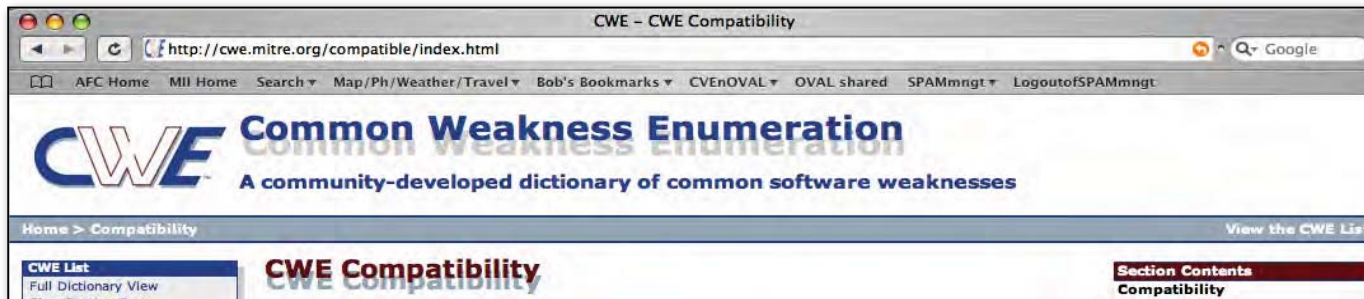
Refining Software Vulnerability Analysis Under ISO/IEC 15408 and ISO/IEC 18045



- The way how the CAPEC and related CWE taxonomies are to be used by the developer, which needs to consider and provide sufficient and effective mitigation to all applicable attacks and weaknesses.
- The way how the CAPEC and related CWE taxonomies are to be used by the evaluator, which needs to consider all the applicable attack patterns and be able to exploit all the related software weaknesses while performing the subsequent AVA_VAN activities.
- How incomplete entries from the CAPEC are to be addressed during an evaluation.
- How to incorporate to the evaluation attacks and weaknesses not included in the CAPEC.

CWE Compatibility & Effectiveness Program

(launched Feb 2007)



Organizations Participating

All organizations participating in the CWE Compatibility and Effectiveness Program are listed below, including those with CWE-Compatible Products and Services and those with Declarations to Be CWE-Compatible.

Products are listed alphabetically by organization name:

cwe.mitre.org/compatible/

TOTALS

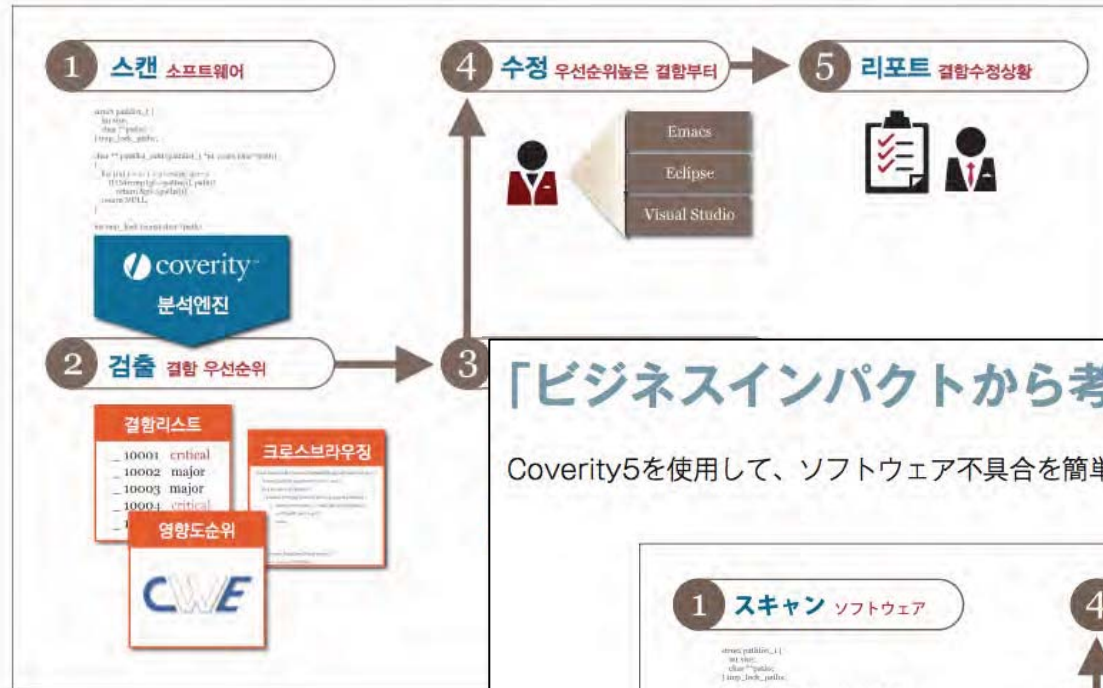
Organizations Participating: 31
Products & Services: 53

December 29, 2006



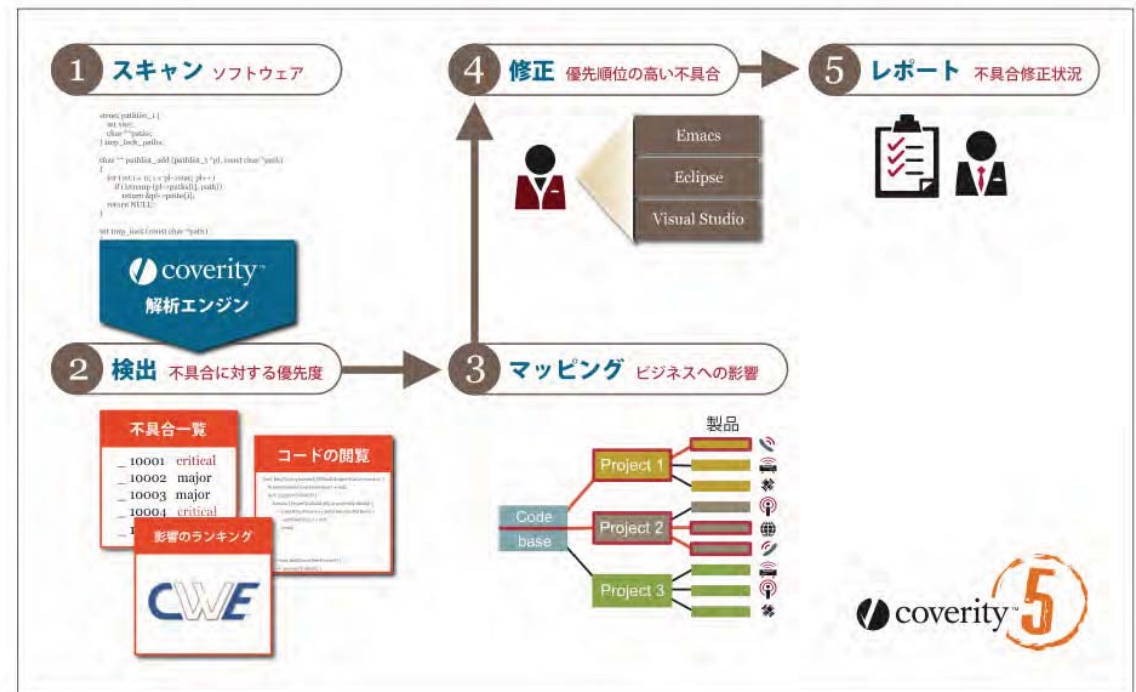
[비즈니스 임팩트를 줄여주는 새로운 품질 관리 방법론]

y5를 사용하여, 소프트웨어 결함을 없애는 5가지 스텝은 아래와 같습니다.



「ビジネスインパクトから考える新しい品質管理」

Coverity5를 사용하여, 소프트웨어 불합을 간단히 제거하는 5단계는以下の通りです。



Korean

Japanese

CWE Coverage – Implemented...

Coverity Data Sheet



Coverity Coverage for Common Weakness Enumeration (CWE): Java

CWE ID	Coverity Static Analysis Checker
171	BAD_EQ
252	CHECKED_RETURN
366	GUARDED_BY_VIOLATION INDIRECT_GUARDED_BY_VIOLATION NON_STATIC_GUARDING_STATIC VOLATILE_ATOMICITY
382	DC.CODING_STYLE BAD_OVERRIDE DC.EXPLICIT_DEPRECATED DC.GC MUTABLE_COMPARISON MUTABLE_HASHCODE
398	

Coverity Data Sheet



Coverity Coverage For Common Weakness Enumeration (CWE): C/C++

CWE ID	Coverity Static Analysis Checker	Checker Description	Type of Security Risk
	TAINTED_SCALAR	Use of untrusted scalar value	Arbitrary control flow
		Untrusted value as an argument	Arbitrary control of a resource
		Use of untrusted value	Arbitrary code execution
		Use of untrusted string value	Arbitrary code execution
		User pointer dereference	Arbitrary code execution
		Out-of-bounds access	Arbitrary code execution
		Stray pointer arithmetic	Arbitrary code execution
		COM bad conversion to BSTRT	Arbitrary code execution
		Overflowed array index write	Arbitrary code execution
		Overflowed pointer write	Arbitrary code execution
		Using invalid iterator	Arbitrary code execution
		Iterator container mismatch	Arbitrary code execution
		Splice iterator mismatch	Arbitrary code execution
		Allocation size error	Arbitrary code execution
		Out-of-bounds access	Arbitrary code execution
		Out-of-bounds write	Arbitrary code execution
		Out-of-bounds access	Arbitrary code execution
		Out-of-bounds write	Arbitrary code execution
		Argument cannot be negative	Arbitrary code execution
		Copy into fixed size buffer	Arbitrary code execution
		Destination buffer too small	Arbitrary code execution
		Possible buffer overflow	Arbitrary code execution
		Allocation too small for type	Arbitrary code execution
		Buffer overflow	Arbitrary code execution
		Copy into fixed size buffer	Arbitrary code execution
		Destination buffer too small	Arbitrary code execution
		Unbounded source buffer	Arbitrary code execution

CWE IDs mapped to Klocwork Java issue types - current

<http://www.klocwork.com/products/documentation/current...>

CWE IDs mapped to Klocwork Java issue types

From current

CWE IDs mapped to Klocwork Java issue types

See also Detected Java Issues.

CWE IDs mapped to Klocwork C and C++ issue types/ja - ...

<http://www.klocwork.com/products/documentation/current...>

CWE IDs mapped to Klocwork C and C++ issue types/ja

From current

< CWE IDs mapped to Klocwork C and C++ issue types

CWE IDs mapped to Klocwork C and C++ issue types/ja

その他の情報 Detected C and C++ Issues.

CWE ID	説明
20 (http://cwe.mitre.org/data/definitions/20.html)	ABV.TAINTED 未検証入力によるバッファ オーバーフロー SV.TAINTED.GENERIC 未検証文字列データの使用 SV.TAINTED.ALLOC_SIZE メモリ割り当てにおける未検証の整数の使用 SV.TAINTED.CALLINDEX.ACCESS =関数呼び出しにおける未検証整数の配列インデックスとしての使用
22 (http://cwe.mitre.org/data/definitions/22.html)	SV.CUDS.MISSING_ABSOLUTE_PATH ファイルのロードでの絶対パスの不使用
73 (http://cwe.mitre.org/data/definitions/73.html)	SV.CUDS.MISSING_ABSOLUTE_PATH ファイルのロードでの絶対パスの不使用
74 (http://cwe.mitre.org/data/definitions/74.html)	SV.TAINTED.INJECTION コマンド インジェクション
77 (http://cwe.mitre.org/data/definitions/77.html)	SV.CODE_INJECTION.SHELL_EXEC シェル実行へのコマンド インジェクション
78 (http://cwe.mitre.org/data/definitions/78.html)	NNTS.TAINTED 未検証ユーザ入力があるバッファ オーバーフロー - 非 NULL 終端文字列 SV.TAINTED.INJECTION コマンド インジェクション
88 (http://cwe.mitre.org/data/definitions/88.html)	SV.TAINTED.INJECTION コマンド インジェクション NNTS.TAINTED 未検証ユーザ入力があるバッファ オーバーフロー

1 of 7

2/26/11 10:34 AM



www.cenzic.com | (866) 4-CENZIC (866-423-6942)

Cenzic Product Suite is CWE Compatible

Cenzic Hailstorm Enterprise ARC, Cenzic Hailstorm Professional and Cenzic ClickToSecure are compatible with the CWE standard or Common Weakness Enumeration as maintained by Mitre Corporation. Web security assessment results from the Hailstorm product suite are mapped to the relevant CWE ID's providing users with additional information to classify and describe common weaknesses found in Web applications.

For additional details on CWE, please visit: <http://cwe.mitre.org/index.html>

The following is a mapping between Cenzic's SmartAttacks and CWE ID's:

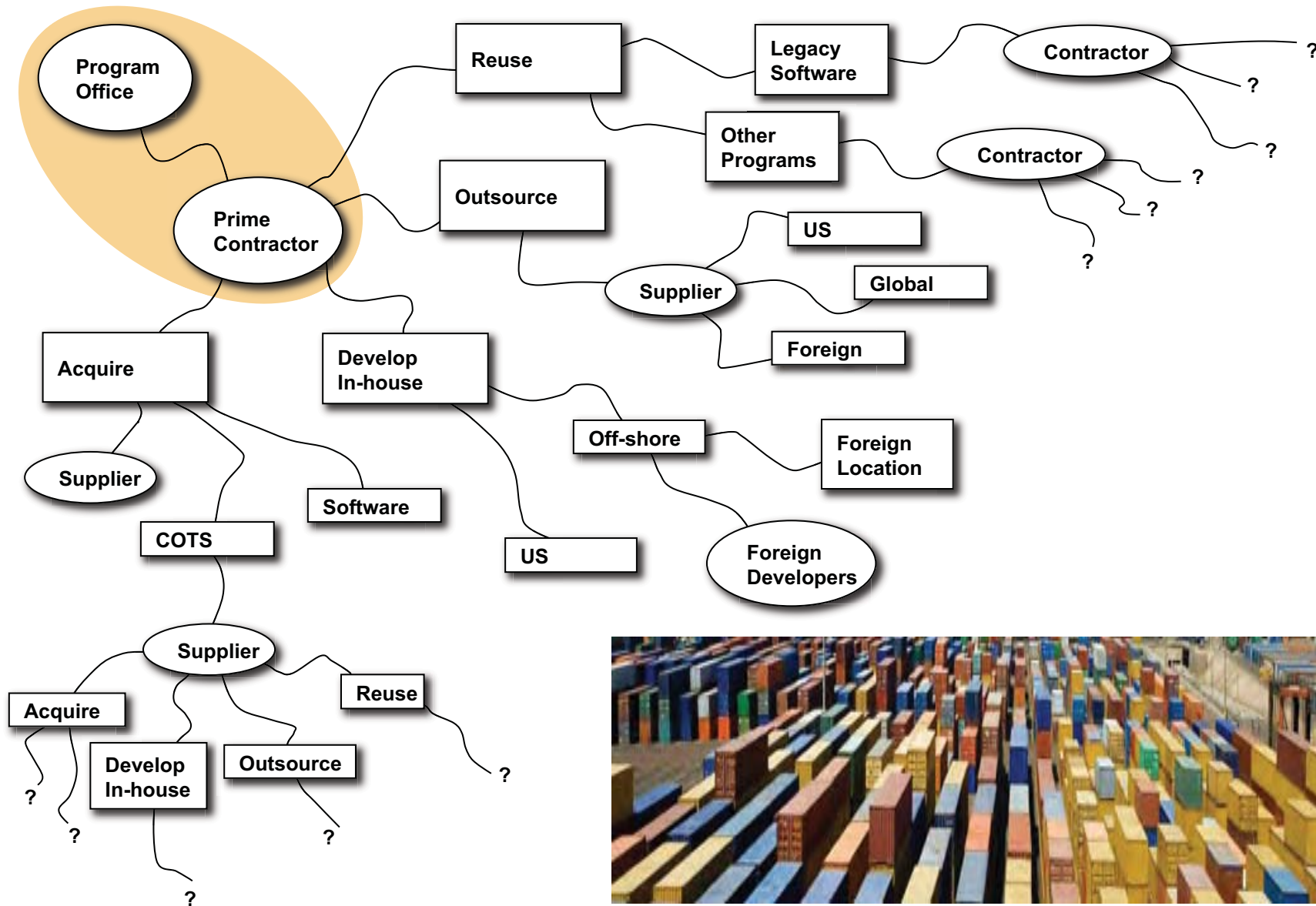
	Cenzic SmartAttack Name	CWE ID/s
1	Application Exception	CWE-388: Error Handling
2	Application Exception (WS)	CWE-388: Error Handling
3	Application Path Disclosure	CWE-200: Information Leak (rough match)
4	Authentication Bypass	CWE-89: Failure to Sanitize Data into SQL Queries (aka 'SQL Injection') (rough match)
5	Authorization Boundary	CWE-285: Missing or Inconsistent Access Control, CWE-425: Direct Request ('Forced Browsing')
6	Blind SQL Injection	CWE-89: Failure to Sanitize Data into SQL Queries (aka 'SQL Injection')
7	Blind SQL Injection (WS)	CWE-89: Failure to Sanitize Data into SQL Queries (aka 'SQL Injection')
8	Browse HTTP from HTTPS List	CWE-200: Information Leak
9	Brute Force Login	CWE-521: Weak Password Requirements
10	Buffer Overflow	CWE-120: Unbounded Transfer ('Classic Buffer Overflow')
11	Buffer Overflow (WS)	CWE-120: Unbounded Transfer ('Classic Buffer Overflow')
12	Check Basic Auth over HTTP	CWE-200: Information Leak
13	Check HTTP Methods	CWE-650: Trusting HTTP Permission Methods on the Server Side

Cenzic CWE Brochure | October 2009

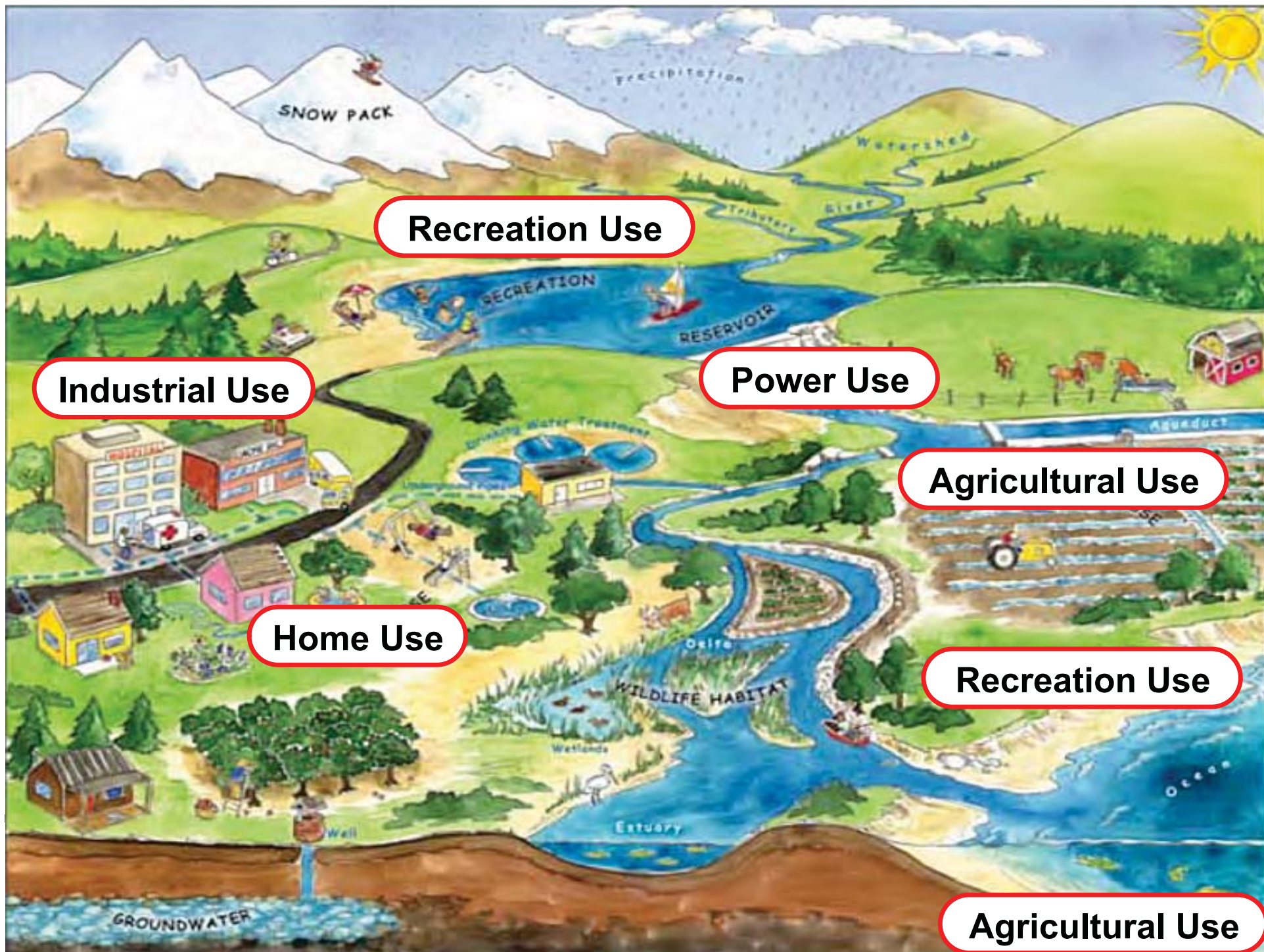
Company Confidential
Cenzic, Hailstorm and ClickToSecure are registered trademarks of Cenzic, Inc.
The Cenzic logo, Hailstorm Enterprise ARC, and ClickToSecure are trademarks of Cenzic, Inc.
© 2009 Cenzic, Inc. All rights reserved.

1

The Software Supply Chain



* “Scope of Supplier Expansion and Foreign Involvement” graphic in DACS www.softwaretchnews.com Secure Software Engineering, July 2005 article “Software Development Security: A Risk Management Perspective” synopsis of May 2004 GAO-04-678 report “Defense Acquisition: Knowledge of Software Suppliers Needed to Manage Risks”



Recreation Use

Industrial Use

Power Use

Agricultural Use

Home Use

Recreation Use

Agricultural Use

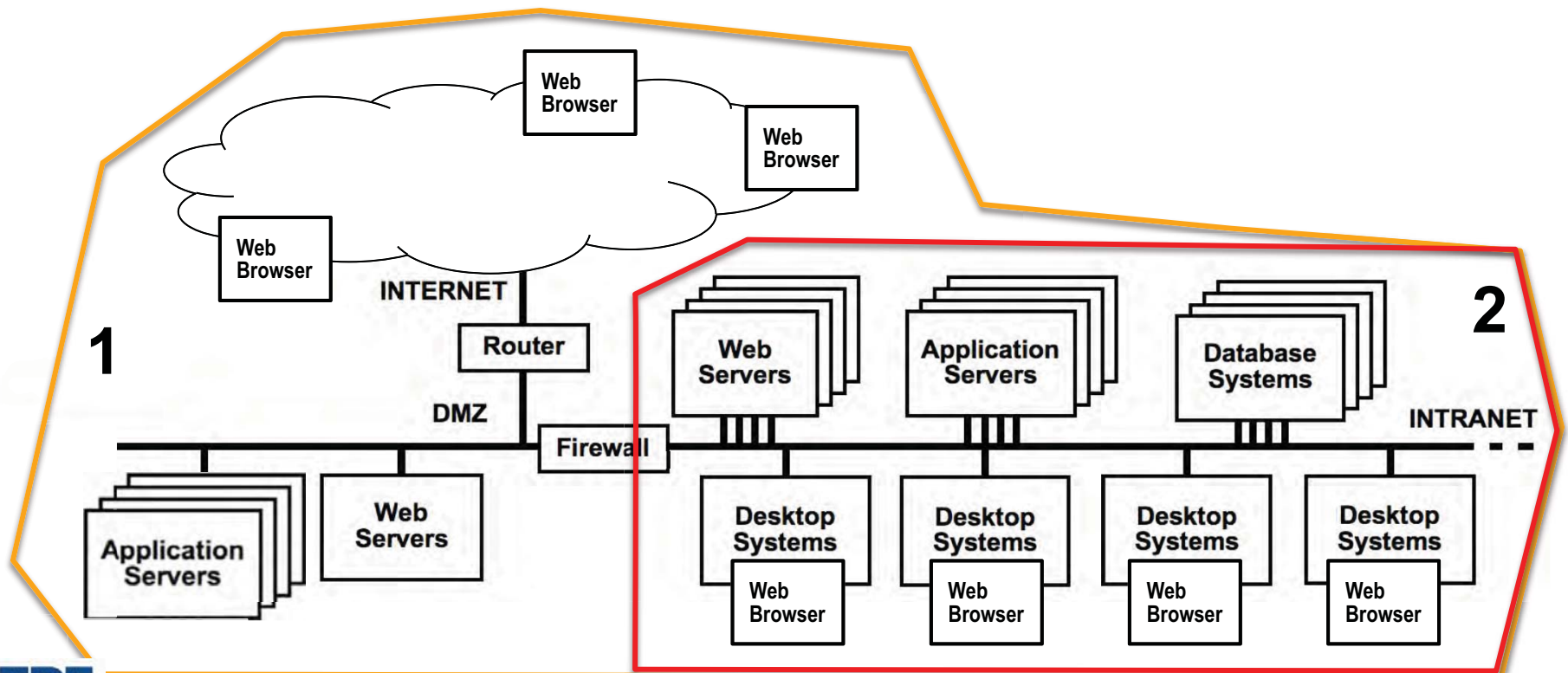
Scoring Weaknesses Based on Context

Archetypes:

- Web Browser User Interface
- Web Servers
- Application Servers
- Database Systems
- Desktop Systems
- SSL

Vignettes:

1. Web-based Retail Provider
2. Intranet resident health records management system of hospital



Domain Name	Description
E-Commerce	The use of the Internet or other computer networks for the sale of products and services, typically using on-line capabilities.
Banking & Finance	Financial services, including banks, stock exchanges, brokers, investment companies, financial advisors, and government regulatory agencies.
Public Health	Health care, medical encoding and billing, patient information/data, critical or emergency care, medical devices (implantable, partially embedded, patient care), drug development and distribution, food processing, clean water treatment and distribution (including dams and processing facilities), etc.
Energy	Smart Grid (electrical network through a large region, using digital technology for monitoring or control), nuclear power stations, oil and gas transmission, etc.
Chemical	Chemical processing and distribution, etc.
Manufacturing	Plants and distribution channels, supply chain, etc.
Shipping & Transportation	Aerospace systems (such as safety-critical ground aviation systems, on-board avionics, etc), shipping systems, rail systems, etc.
National Security	National security systems (including networks and weapon systems), Defense Industrial Base, etc.
Government and Commercial Security	Homeland Security systems, commercial security systems, etc.
Emergency Services	Systems and services that support first responders, incident management and response, law enforcement, and emergency services for citizens, etc.
Telecommunications	Cellular services, land lines, VOIP, cable & fiber networks, etc.
Telecommuting & Teleworking	Support for employees to have remote access to internal business networks and capabilities.
eVoting	Electronic voting systems, as used within state-run elections, shareholder meetings, etc.

Technology Group	Archetypes/Description
Web Applications	Web browser, web-server, web-based applications and services, etc.
Industrial Control Systems	SCADA, process control system, etc.
Real-time, Embedded Systems	Embedded Device, Programmable logic controller, implanted medical devices, avionics package.
End-point Computing Devices	Smart phone, laptop, personal digital assistant (PDA), and other remote devices that leave the enterprise and/or connect remotely to the enterprise.
Cloud Computing	Hosted applications or capabilities provided over the Internet, including Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure as a Service (IaaS).
Operating Systems	General-purpose OS, virtualized OS, Real-time operating system (RTOS), hypervisor, microkernel.
Enterprise Desktop Applications/Systems	Office products such as word processing, spreadsheets, project management, etc.

Vignettes – Technology Groups & Business/Mission Domains

Technology Groups	Business/Mission Domains															
	e-Commerce	Banking & Finance	Energy (i.e., SmartGrid, nuclear power, oil/gas transmission)	Chemical	Manufacturing	Shipping & Transportation (i.e., rail, Freight, ships, airlines, aerospace, postal)	National Defense (i.e., weapon systems, Intel networks, defense industrial base)	Homeland Security (CBP, Coast Guard, Secret Service, TSA, etc.)	Government (other than Nat'l Def & HS)	Emergency Services (law enforcement, Incident response, security services, etc.)	Public Health	Food & Water	Telecommunications	Teleworking	e-Voting	
Web Applications																
Real-Time Embedded Systems																
Control Systems																
End-Point Computing Devices																
Database & Storage Sys																
Operating Systems																
Identity Mngt Systems																
Enterprise Sys Apps																
Cloud Computing																

Common Vignette for Domain

Vignette for Domain/ Tech Gp

Common Vignette for Technology Group

Common Vignette for Technology Group

Vignette for Domain/ Tech Gp

Common Vignette for Domain

Vignette for Domain/Tech Gp

Common Vignette for Technology Group

Common Vignette for Technology Group

Vignette for Domain/Tech Gp

CWRAF-Level Technical Impacts

- 1. Modify data**
- 2. Read data**
- 3. DoS: unreliable execution**
- 4. DoS: resource consumption**
- 5. Execute unauthorized code or commands**
- 6. Gain privileges / assume identity**
- 7. Bypass protection mechanism**
- 8. Hide activities**

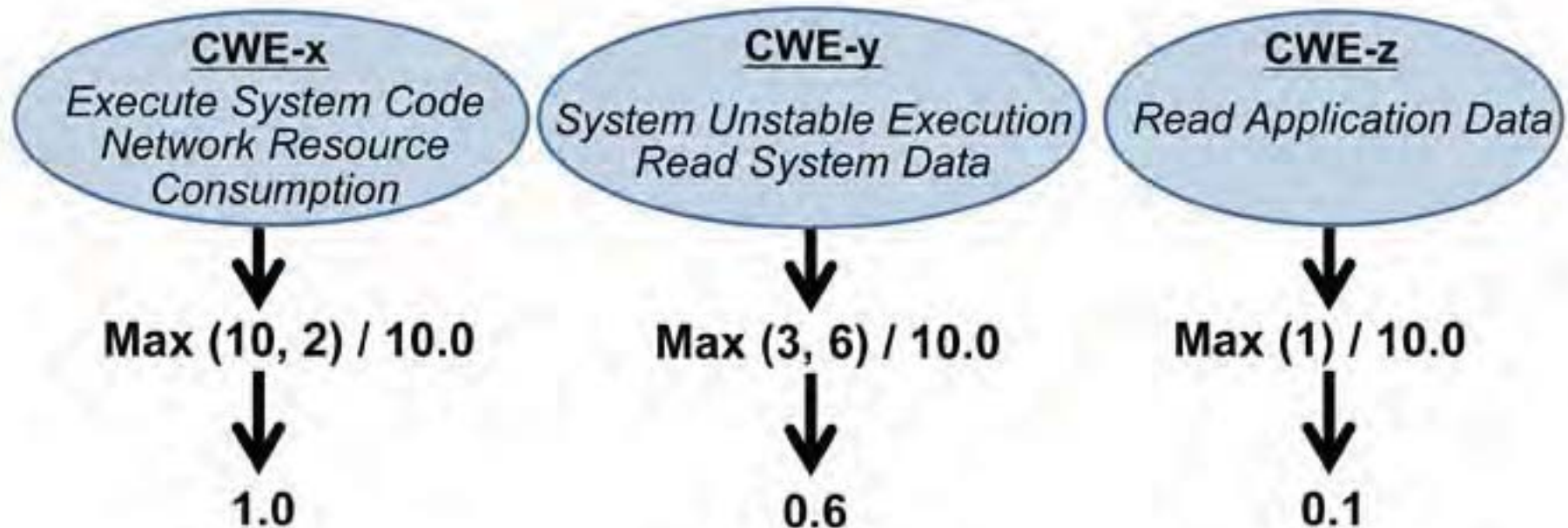
Technical Impact Scorecard

- Links business value with the technical impact of weakness exploitation
- Stays away from technical details of individual weaknesses
- Operates within the context of a vignette

Calculating CWSS Impact Weights

10 – Execute System Code
6 – Read System Data
3 – System Unstable Execution
2 – Network Resource consumption
1 – Read Application Data

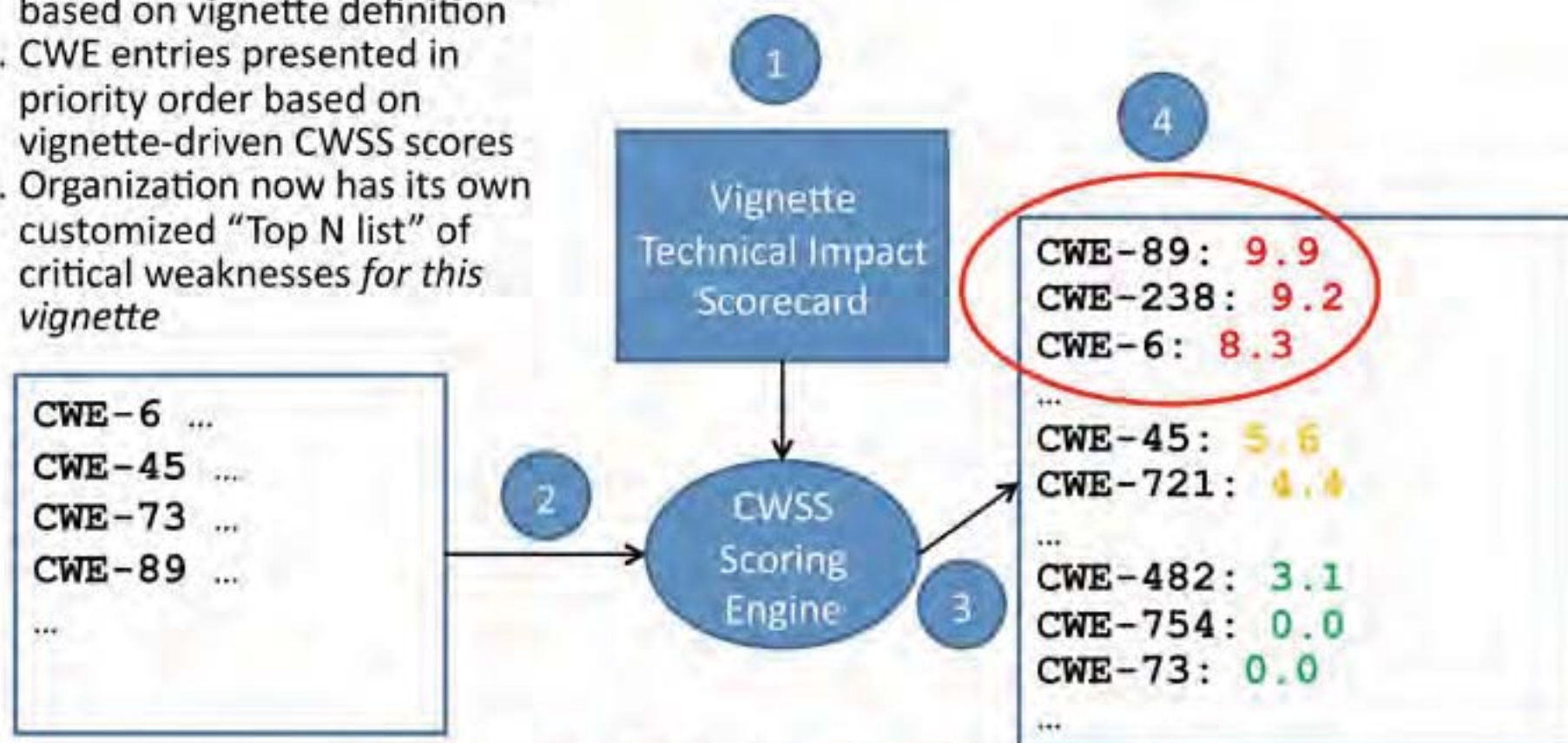
*Technical
Impact
Scorecard*



Scoring Relevant Weaknesses using CWSS

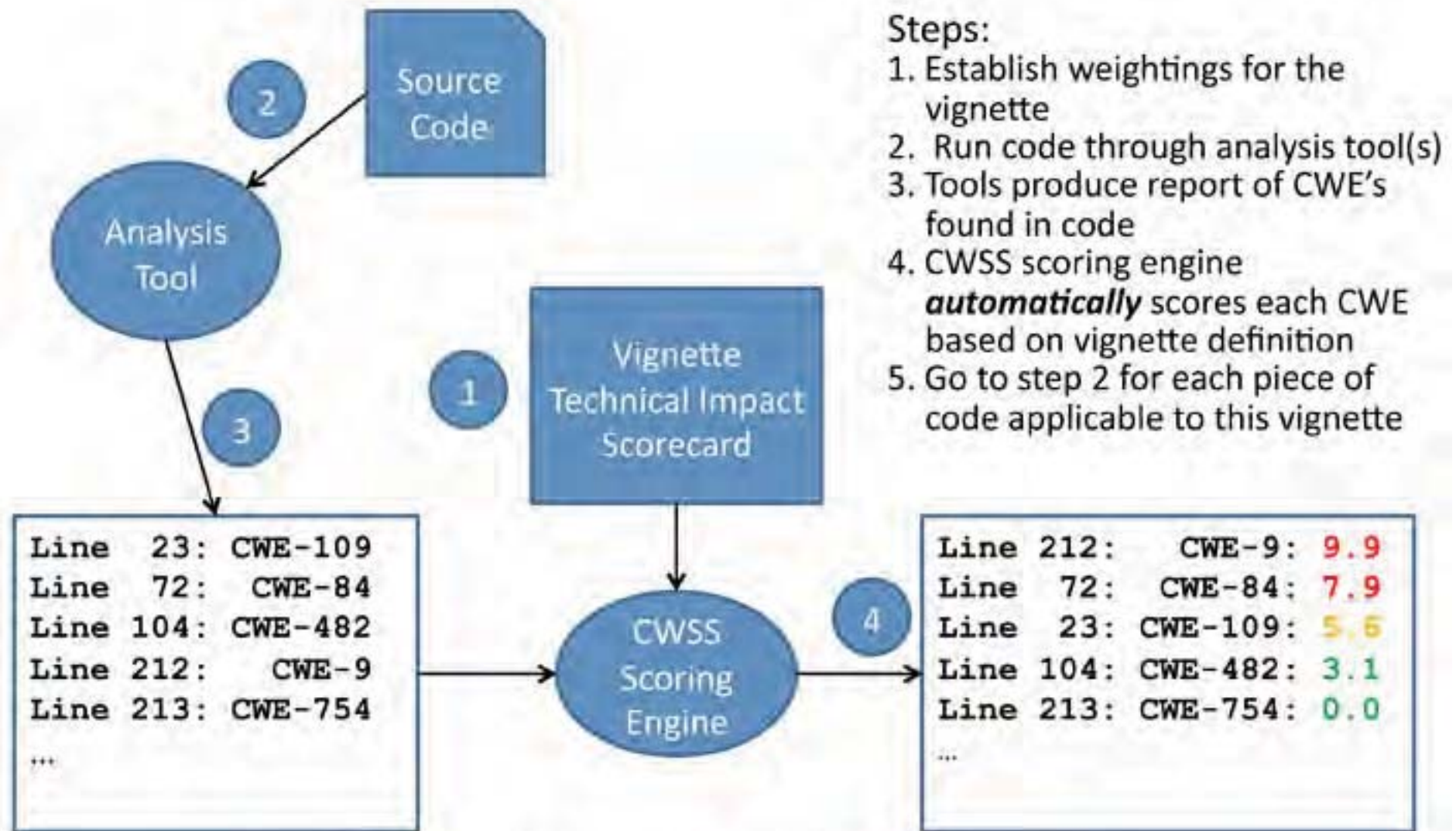
Steps:

1. Establish weightings for the vignette
2. CWSS scoring engine processes each relevant CWE entry and *automatically* scores the entry based on vignette definition
3. CWE entries presented in priority order based on vignette-driven CWSS scores
4. Organization now has its own customized “Top N list” of critical weaknesses *for this vignette*



Step 1 is only done once – the rest is automatic

Scoring Weaknesses Discovered in Code using CWSS



Exercise: Build a Vignette → Create Technical Scorecard → Get “our” Top 25

CWRAF-Level Technical Impacts

Weights

Impacts

1↔10	Modify data
1↔10	Read data
1↔10	DoS: unreliable execution
1↔10	DoS: resource consumption
1↔10	Execute unauthorized code or commands
1↔10	Gain privileges / assume identity
1↔10	Bypass protection mechanism
1↔10	Hide activities



VIGNETTE

CWSS for a Technology Group

50%	Web Vignette 1 ... TI(1), TI(2), TI(3),...	Top N List 1
10%	Web Vignette 2 ... TI(1), TI(2), TI(3),...	Top N List 2
10%	Web Vignette 3 ... TI(1), TI(2), TI(3),...	Top N List 3
10%	Web Vignette 4 ... TI(1), TI(2), TI(3),...	Top N List 4
15%	Web Vignette 5 ... TI(1), TI(2), TI(3),...	Top N List 5
15%	Web Vignette 6 ... TI(1), TI(2), TI(3),...	Top N List 6

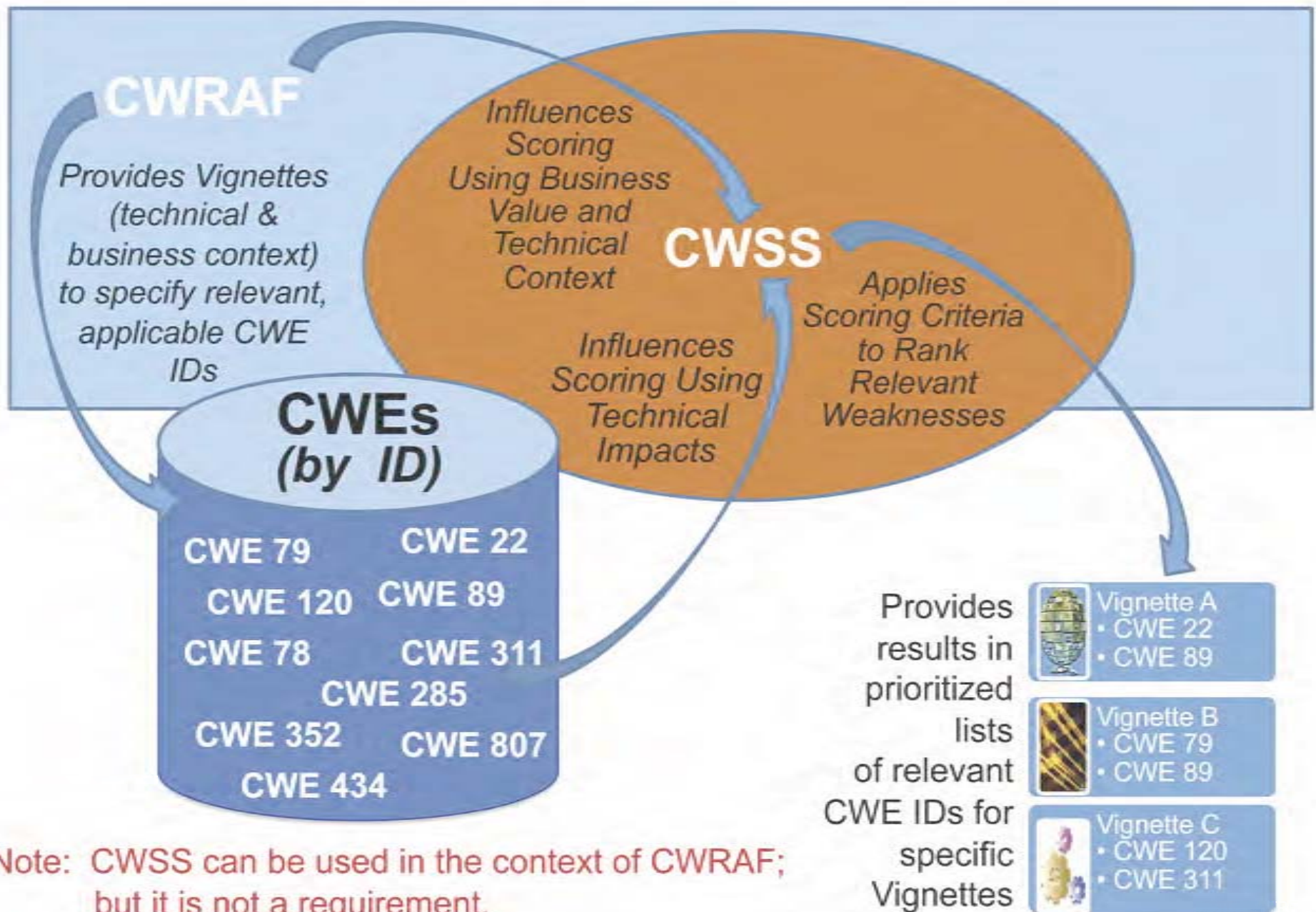
Web Application Technology Group

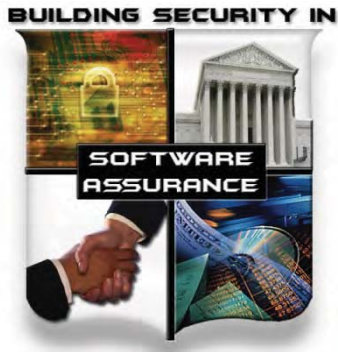
Top 10 List

CWE Top 10 List for Web Applications can be used to:

- **Identify skill and training needs for your web team**
- **Include in T's & C's for contracting for web development**
- **Identify tool capability needs to support web assessment**

Relationships between CWRAF, CWSS, and CWE





Contact Info



cwss@mitre.org

cwe@mitre.org



© 2011 MITRE